Departamento de Tecnología Electrónica

PROYECTO FIN DE CARRERA

INTERFAZ GRÁFICA EN MATLAB PARA EL ANÁLISIS DE DESCARGAS PARCIALES EN DETECCIÓN ELÉCTRICA Y ACÚSTICA

Autor: Pablo Grandas Aguado
Directores: José Antonio García Souto
Jesús Rubio Serrano

Leganés, Noviembre de 2011

Título: INTERFAZ GRÁFICA EN MATLAB PARA EL ANÁLISIS DE DESCARGAS PARCIALES EN DETECCIÓN ELÉCTRICA Y ACÚSTICA
Autor: Pablo Grandas Aguado
Directores: José Antonio García Souto y Jesús Rubio Serrano

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 18 de Noviembre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco a José Antonio García Souto y Jesús Rubio Serrano del departamento de Tecnología Electrónica por la oportunidad y la ayuda que he recibido.

A mis padres Paula y Francisco por la motivación y la paciencia que han tenido durante todos estos años (y los que quedan).

A mi familia a la que no he visto tanto como me gustaría durante el desarrollo del proyecto. Especialmente a mi abuelo Francisco.

A Elena por el apoyo durante todo el desarrollo del proyecto y redacción de la memoria, la paciencia y todo el tiempo que compartimos juntos.

A Ivaneme y Kumpel por el servicio técnico de urgencia y los ratos de relax, necesarios en todos los aspectos de la vida.

A todos los que han estado cerca durante la carrera, tanto de la universidad como de fuera de ella.

Resumen

Las descargas parciales son procesos eléctricos que provocan una gran cantidad de "pequeños cortocircuitos" dentro del aislante de máquinas eléctricas. Estas descargas, aunque suelen ser de pequeña magnitud (cientos de pC), por su repetición y su persistencia son las responsables de la degradación del aislante y un síntoma del estado de envejecimiento del mismo.

En el presente proyecto se ha desarrollado una interfaz gráfica de usuario en lenguaje MATLAB que facilita a un experto el análisis de señales acústicas y eléctricas procedentes de descargas parciales. Adicionalmente, la aplicación desarrollada asistiría a un operador con formación específica en el diagnóstico basado en medidas acústicas y eléctricas de descargas parciales.

Para el diseño de la interfaz gráfica se parte de un conjunto de funciones matemáticas de procesamiento desarrolladas previamente por un experto y de un conjunto de señales adquiridas en experimentos de laboratorio. Se ha estudiado el funcionamiento de cada una de ellas y las interrelaciones para diseñar el secuenciamiento del proceso de detección y análisis de descargas parciales y la estructura de la base de datos de resultados. El resultado es una aplicación con interfaz gráfica que se estructura en una serie de herramientas de análisis específico.

La aplicación desarrollada (*PDtool*) permite gestionar las bases de datos de señales y la ejecución de funciones de detección y análisis de descargas parciales. En la presente versión la información proviene de señales adquiridas previamente sobre las que se aplica el post-procesamiento. La presentación de resultados se realiza automáticamente siendo seleccionados los más significativos para cada tipo de análisis. Permite procesos autoconfigurados de detección y localización para usuarios menos familiarizados con los algoritmos de procesamiento de señal, así como que usuarios expertos puedan realizar configuraciones a un nivel más avanzado.

Palabras clave: Interfaz de usuario GUI (Guided User Interface), diagnóstico de descargas parciales, base de datos de descargas parciales.

Abstract

Partial discharges are electrical processes that cause a great amount of “small short-circuits” inside the insulation of electrical machines. These discharges are the responsible of the degradation in the insulation and a symptom of the aging of the insulation because of their repetition and persistence, although they have small magnitude (hundreds of pC).

In this project, a graphical user interface in MATLAB has been developed, that eases the analysis of electric and acoustic signals from partial discharges to an expert technician. In addition, the developed application will assist an operator, with specific training, in the diagnosis based on acoustic and electric measurements of partial discharges.

The starting point is a set of mathematical processing functions, previously developed by an expert, and a set of signals acquired in laboratory experiments. The action of each function and the interrelationships between them has been studied in order to design the sequence of the process of detection and analysis of partial discharge and the structure of the database results. The result is a graphic user interface application structured in a series of specific analysis tools.

The developed application (*PDtool*) manages the databases of the signals and the execution of functions of detection and analysis of partial discharges. In the current version, the information comes from a previously acquired set of signals in which a post-processing is applied. The presentation of the results shows, automatically, the most significant data for each type of analysis. The application has two levels of configuration: basic and advance. Basic configuration allows the detection and the localization for users without deep knowledge of the signal processing algorithms. Advanced configuration allows deep configuration of the functions for expert users.

Keywords: Guided User Interface (GUI), partial discharge analysis, partial discharge database.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Motivación

Las descargas parciales (DP) son procesos eléctricos que provocan una gran cantidad de "pequeños cortocircuitos" dentro del aislante del que están construidos los equipos eléctricos. Estas descargas, aunque suelen ser de pequeña magnitud (cientos de pC), por su repetición y su persistencia son las responsables de la degradación del aislante y a su vez un síntoma del estado de dicha degradación.

Para estudiar las DP se ha partido de un experimento de generación de DP que cumple con la norma establecida [IEC 60270]. Este experimento además consta de una cuba llena de aceite donde se sumerge el dispositivo donde se van a generar las DP a detectar.

Para detectar las señales emitidas se usa un sensor eléctrico que mide el transitorio de carga generado por la DP y adicionalmente un conjunto de sensores acústicos ubicados en diferentes posiciones en las paredes de la cuba con el objeto de poder localizar el lugar de generación de la DP. Para medir y registrar las señales, se cuenta con un sistema de adquisición basado en dos módulos CS328A de Cleverscope Ltd, sincronizados para obtener 4 canales de 14 bits ADC con una frecuencia de muestreo de 100 MSps y 2 millones de muestras guardadas por cada canal. Por último, las series temporales son almacenadas y posteriormente procesadas mediante una serie de funciones escritas en MATLAB con el objeto de detectar, analizar, caracterizar y localizar las DP generadas en el experimento.

En la Figura 1 se muestra el procedimiento de trabajo utilizado por un experto para generar, adquirir y procesar series temporales de DP.

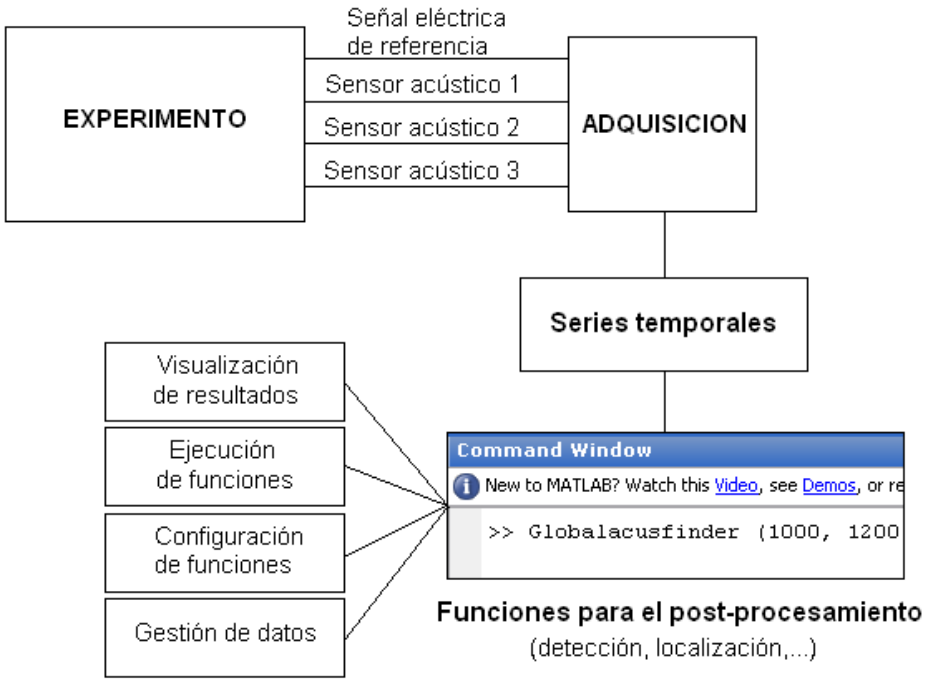


Figura 1.- Esquema del sistema inicial de procesamiento para la detección y análisis de descargas parciales

Para la detección e identificación de descargas parciales se necesita un post-procesamiento de las series temporales adquiridas. Para ello, un experto ha diseñado un conjunto de funciones encargadas de la detección, localización y análisis de descargas parciales. El experto utiliza estas funciones en el entorno de MATLAB ejecutándolas en la línea de comandos, lo que hace que su utilización se vea limitada únicamente a usuarios que conozcan su funcionamiento interno ya que estas funciones necesitan unos argumentos de entrada concretos y una configuración específica para generar los archivos correspondientes de forma correcta. Para un correcto uso de las funciones, es necesario ejecutarlas de forma secuencial y gestionar los datos generados por las mismas. Todas las funciones están diseñadas para funcionar de forma independiente. Es decir, son unidades de procesamiento individual pero no existe una aplicación que agrupe y secuencie todo este conjunto de funciones.

Debido a la dificultad en el manejo del conjunto de funciones, en el presente proyecto se ha desarrollado una aplicación en lenguaje MATLAB que haciendo uso de dichas funciones facilite a un experto el análisis de señales acústicas y eléctricas procedentes de descargas parciales. Adicionalmente, la aplicación desarrollada asistiría a un operador con formación específica en el diagnóstico basado en medidas acústicas y eléctricas de descargas parciales.

1.2 Objetivos

El objetivo fundamental de este proyecto es el diseño de una interfaz gráfica de usuario (Figura 2) en entorno MATLAB que facilite al usuario el procesamiento de las señales adquiridas en el experimento de detección electro-acústica de descargas parciales mediante el conjunto de funciones suministrado por un experto destinadas a la detección y localización de descargas parciales.

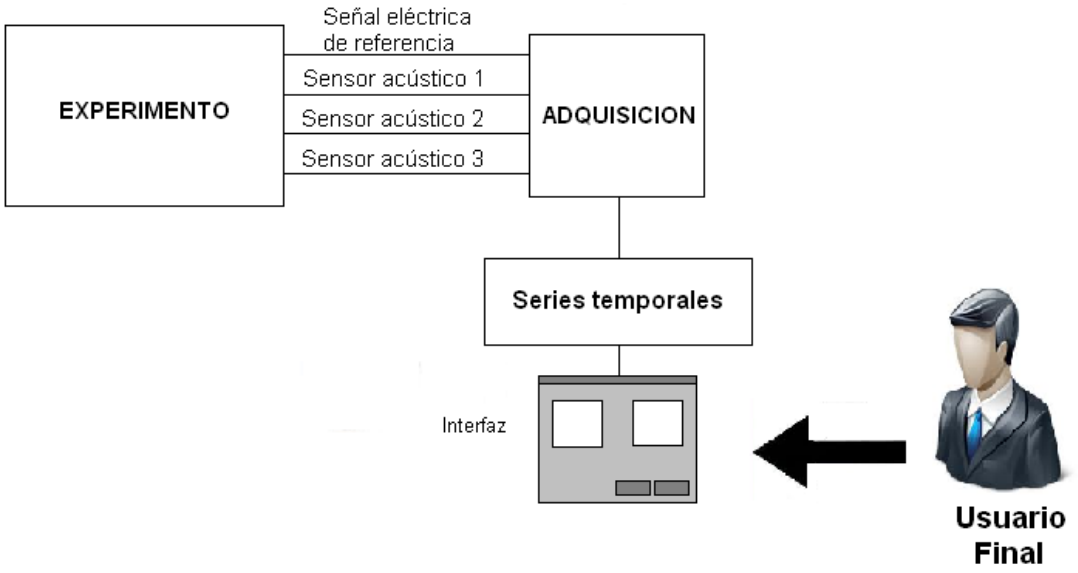


Figura 2.- Implantación de la interfaz en el sistema de procesamiento para la detección y análisis de descargas parciales

La aplicación deberá gestionar las funciones así como la información necesaria para su correcto funcionamiento, sustituyendo un conjunto de funciones y de datos por una aplicación compacta que conserva la funcionalidad original del conjunto de funciones (Figura 3).

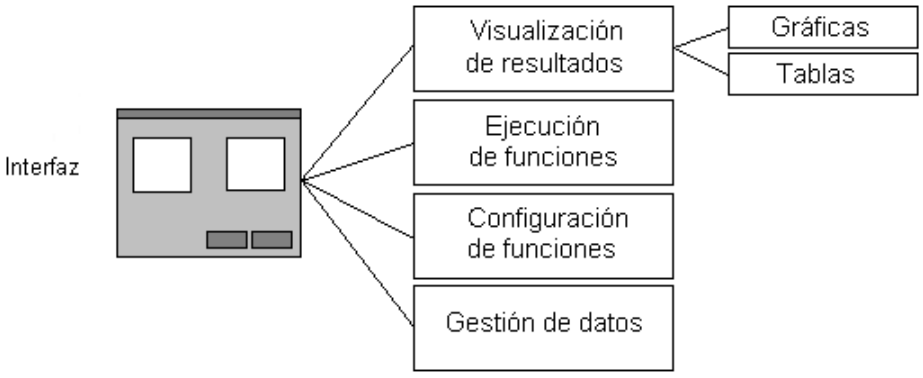


Figura 3.- Funciones que desempeña la interfaz

A partir de este objetivo principal se definen los siguientes objetivos parciales:

- La aplicación debe organizar secuencialmente las funciones de procesamiento de las señales y gestión de los datos para que guarde un orden lógico a la hora de realizar una detección de descargas parciales. El usuario no necesitará conocer las funciones que se ejecutan con la interfaz para realizar un diagnóstico.

- La aplicación debe ser la encargada de gestionar las bases de datos de descargas parciales para que las funciones dispongan de la información necesaria para su correcto funcionamiento.

- La interfaz debe mostrar los resultados obtenidos tras la detección y análisis de descargas parciales de forma gráfica, mediante gráficas y tablas.

- La interfaz debe permitir una configuración de todas las funciones utilizadas. Para ello se creará una configuración por defecto para que usuarios con formación específica puedan realizar el procesado y existirá la posibilidad de configurar aspectos más avanzados para un análisis de un usuario experto.

- La interfaz debe tener una estructura clara y ordenada para que un usuario poco familiarizado con la detección y análisis de descargas parciales pueda obtener información suficiente para sacar unas primeras conclusiones. La interfaz será la encargada de solicitar la información necesaria para que se pueda realizar la detección de descargas parciales.

- La aplicación debe estar basada únicamente en ventanas, evitando utilizar la línea de comandos. Para ello es necesaria la creación de un sistema de ventanas en el que se deberá mostrar la información asociada a cada parte del proceso de detección y análisis de descargas parciales.

- Desarrollo de un sistema de selección gráfica de datos. Esto es importante ya que permite al usuario interactuar de forma gráfica con cierto tipo de datos. De otra manera tendría que interactuar por medio de los comandos de MATLAB o herramientas presentes en el entorno MATLAB. Integrándolo en la herramienta se evita el paso por la pantalla de MATLAB.

1.3 Fases del desarrollo

Se pueden diferenciar cuatro fases fundamentales para el desarrollo de la herramienta.

- Familiarización con el problema y el entorno de trabajo. En esta primera etapa se estudia de forma básica el experimento para tener una idea global del experimento y así poder aportar ideas propias a la herramienta. También se estudia el entorno en el que se va a desarrollar la interfaz, así como las entradas y salidas de las funciones y el flujo de los datos.

- Fase de entrevistas para definir los requisitos de la herramienta. Tras familiarizarse con el entorno de trabajo, el siguiente paso es definir los requisitos técnicos y de diseño de la herramienta mediante entrevistas con el usuario final. Tras acordar unos requisitos en cuanto al diseño, las especificaciones básicas y descartar objetivos inalcanzables se pasa a la siguiente fase de desarrollo

- Desarrollo de la herramienta. Se desarrolla la herramienta en el entorno elegido respetando los requisitos acordados en la fase previa. Esta fase es la que más tiempo consume ya que durante la tarea de diseñar y programar surgen problemas que solucionar. A la mitad de esta etapa se debe probar la herramienta con el usuario final en previsión de deficiencias o admitiendo la posibilidad de añadir nuevas ideas propias o por parte del usuario. Se ha optado por realizar primero programas simples y basándose en estos programas ir incrementando su funcionalidad siempre adaptándose a los requisitos técnicos.

- Prueba de la aplicación. Una vez terminada la aplicación se prueba con el usuario final para ver el resultado. Tras esta versión se podrán realizar mejoras sobre el programa final, pero respetando todo el trabajo previo.

1.4 Medios empleados

Para realizar este proyecto se ha utilizado la herramienta de cálculo MATLAB Versión 7.8.0 R2009a, que permite diseñar la interfaz mediante la herramienta GUIDE para crear interfaces GUI (Guided User Interface).

También se dispone de las bases de datos con las señales adquiridas en experimentos electro-acústicos de DP, los patrones con los que se compara la señal para detectar las DP y el conjunto de funciones necesarias para el correcto procesamiento de las señales.

El equipo sobre el que se ha realizado el diseño de la herramienta es un ordenador portátil LG E500 Intel(R) Core(TM)2 Duo CPU (@2.40GHz) y 3GB de RAM (ANEXO 11.1). Para la instalación de MATLAB Versión 7.8.0 R2009a el equipo mínimo es Intel Pentium 4 (1,4 y 1,5 GHz), 512MB de RAM y 680 MB libres en el disco duro (ANEXO 11.2).

1.5 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

El presente capítulo consta de una descripción del problema a tratar, en este caso el diseño de una interfaz para asistir en la detección y el análisis de descargas parciales, y la

motivación que hay para el desarrollo del proyecto. También se detallan los principales objetivos que se pretenden alcanzar con la realización de este proyecto. Adicionalmente se describen las distintas fases de desarrollo del proyecto y los medios con los que se han contado para el desarrollo del mismo.

El segundo capítulo recoge una explicación de cómo trata los datos un experto durante el análisis de DP para visualizar las primeras ideas de cómo orientar el proyecto. Este capítulo contiene una descripción de todas las funciones suministradas por el programador experto en DP, así como una descripción de las señales adquiridas y las distintas bases de datos de descargas parciales. Por último se extraen unas conclusiones en las que se presenta un esquema de las funciones trabajando de forma secuencial. Este será el primer boceto de la aplicación que respaldará la parte gráfica de la interfaz.

El tercer capítulo contiene una descripción de las distintas posibilidades que ofrece MATLAB a la hora de crear interfaces. Se resume el modo de programación de interfaces con la herramienta GUIDE. Se explica también como manejar las bases de datos con archivos de tipo *cell*, en los que se basan las bases de datos suministradas. Por último se presenta la herramienta utilizada para la selección gráfica de datos con algún ejemplo de aplicación.

En el cuarto capítulo se desarrolla la solución adoptada. Se muestra esquemáticamente la estructura de la aplicación desarrollada así como la de los distintos submenús, acompañadas de una descripción del funcionamiento de cada uno de los submenús. También se diseña la parte gráfica de las distintas ventanas.

Tras finalizar el desarrollo se comprueba que la interfaz cumple con los requisitos. En el quinto capítulo se muestra un resumen de las pantallas que se crean durante el funcionamiento del programa. En el sexto capítulo de conclusiones y trabajo futuro se resumen los objetivos alcanzados al finalizar el proyecto y posibles mejoras para siguientes versiones. Por último, en el séptimo capítulo se resume el presupuesto del proyecto en el que se detallan la mano de obra y el equipo utilizado para su desarrollo.

En los anexos se aporta el manual de usuario. También se ha añadido un resumen de las entrevistas con el usuario experto y los primeros bocetos en los que se basó el sistema de ventanas.

Capítulo 2

Detección y localización de descargas parciales

2.1 Introducción

Las DP se definen en la **norma IEC 60270 [1]** como: "descarga eléctrica localizada que sólo atraviesa parcialmente el aislante entre conductores y que puede ocurrir o no cerca de un conductor."

Las descargas parciales (DP) son procesos eléctricos que provocan una gran cantidad de "pequeños cortocircuitos" dentro del sistema aislante de máquinas eléctricas. Estas descargas, aunque suelen ser de pequeña magnitud (cientos de pC), por su repetición y su persistencia son las responsables de la degradación del aislante y un síntoma del estado de degradación del mismo.

Para generar las series temporales emitidas por las descargas parciales, investigadores del Departamento de Tecnología Electrónica y del Departamento de Ingeniería Eléctrica han diseñado una plataforma de ensayo acorde a la norma IEC 60270, que posteriormente se procesan para detectar y localizar la existencia de DP [2],[3].

Un esquema típico de medida de DP se muestra en la Figura 4; en el mismo hay que destacar que se trata de un sistema de adquisición de cuatro canales donde uno de los sensores mide el transitorio de carga eléctrico y los otros tres sensores son para medir las señales que llegan a las paredes de la cuba.

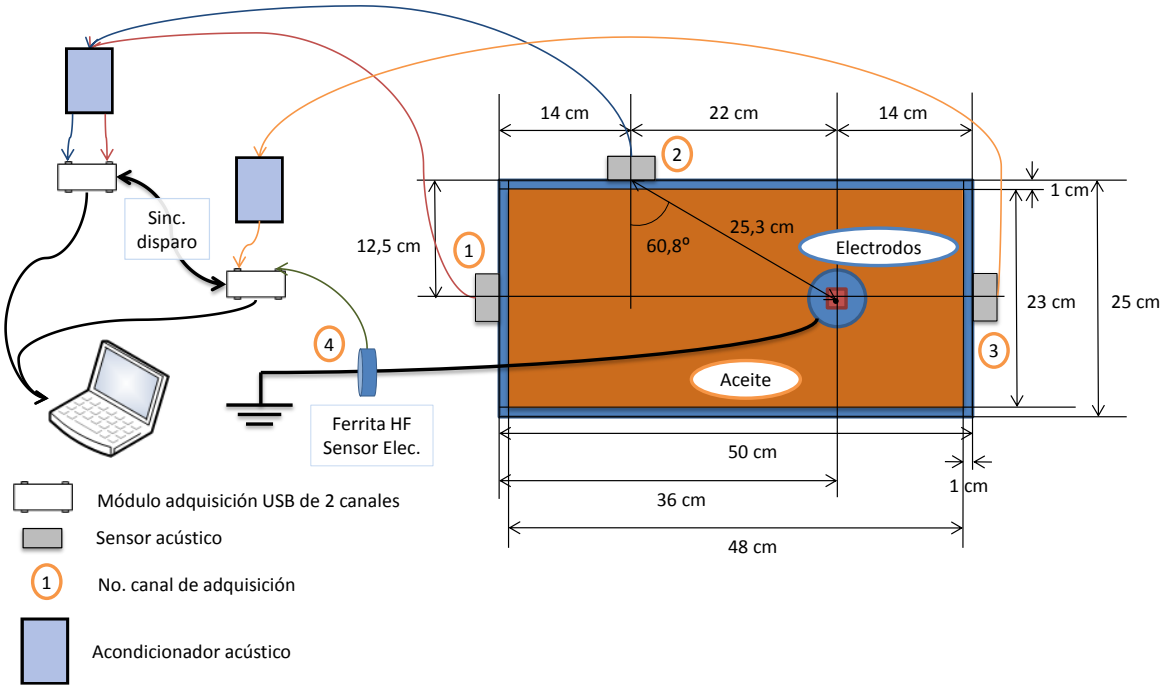


Figura 4.- Esquemático del montaje experimental de descargas parciales[3]

El esquema experimental de la Figura 4 se compone de tres elementos importantes: un sistema de generación de DP, un conjunto de sensores acústicos y eléctricos y un sistema de adquisición de cuatro canales

La parte de generación consiste en una cuba de PMMA llena de aceite donde se sumerge el sistema de generación de DP. Dentro de esta cuba se producirán las descargas parciales.

Al producirse la descarga parcial se producirá un transitorio eléctrico que se detecta con la ferrita HF y una emisión acústica en todas las direcciones que se detecta con un array de sensores acústicos ubicados en la pared de la cuba.

El sistema de adquisición captura y almacena la información que llega a cada uno de los cuatro canales de forma sincronizada. Una vez captadas y almacenadas, el experto le aplicará determinadas funciones para limpiar el ruido la señal y para obtener la información necesaria para la detección y localización de las descargas parciales.

2.2 Flujo de datos típico en experimentos DP

Desde el punto de vista de un experto un análisis de datos típicos para la detección consta de los siguientes puntos [2]:

1. *Detectar* señales temporales generadas por las descargas parciales y que pueden ser detectadas por varios sensores que pueden ser de distinto tipo.
2. *Identificar* las señales que pueden ser generadas por los distintos tipos de descargas parciales: internas, superficiales y corona.

3. *Localizar* el origen de la DP detectada.
4. *Procesar* de forma estadística los eventos detectados para determinar la existencia localizada de defectos en el aislante y sus características.

Para ello, previamente ha desarrollado un conjunto de funciones en MATLAB que combina herramientas de procesamiento de señal (para detectar e identificar eventos en los registros), herramientas de localización (para localizar temporal y espacialmente eventos) y herramientas estadísticas (para dar validez a los resultados, encontrando conjuntos de eventos localizados en una misma región).

La Figura 5 muestra el procedimiento que sigue un experto para detectar, identificar y localizar las DP y que se van a describir en las secciones siguientes.

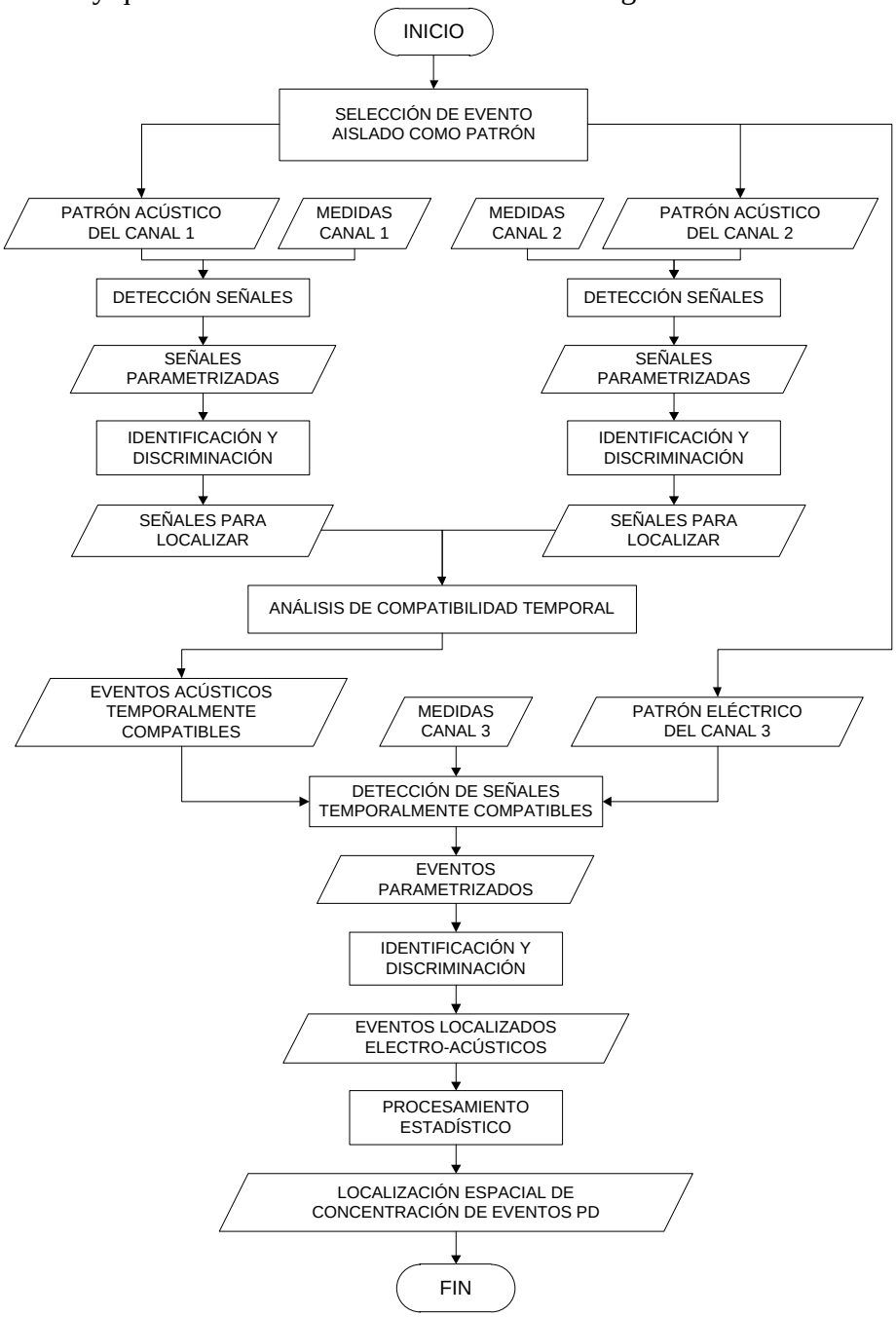


Figura 5.- Proceso de detección y localización de DP [2]

2.2.1 Detección

Para detectar señales en los registros temporales, el primer paso es seleccionar el evento representativo que se desea detectar en las medidas y obtener una forma de onda representativa de la señal DP por canal. En la Figura 6 se muestra un ejemplo de las señales que se han detectado en un experimento de dos sensores acústicos y uno eléctrico.

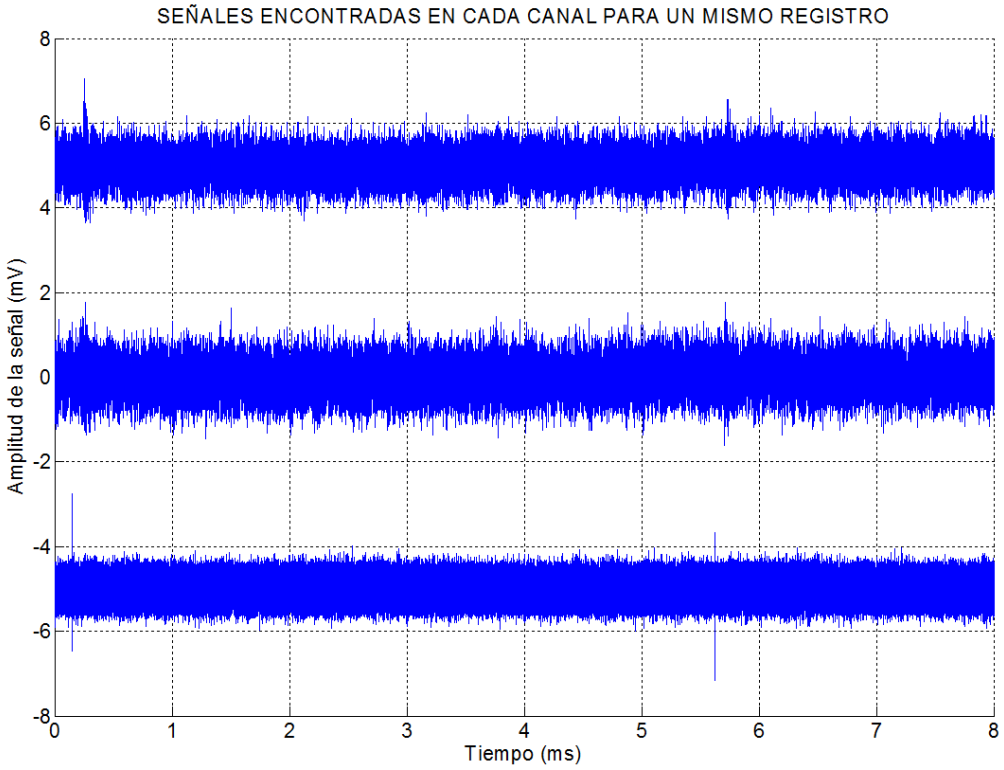


Figura 6.- Adquisición típica del experimento[3]

Aunque el nivel de señal en la figura 6 dificulta la detección de DP en los canales acústicos, es posible identificar el evento basándose en la conjunción de varios parámetros como el retardo entre las distintas señales o su forma de onda. Un ejemplo de un evento representativo de una DP se muestra en la Figura 7.

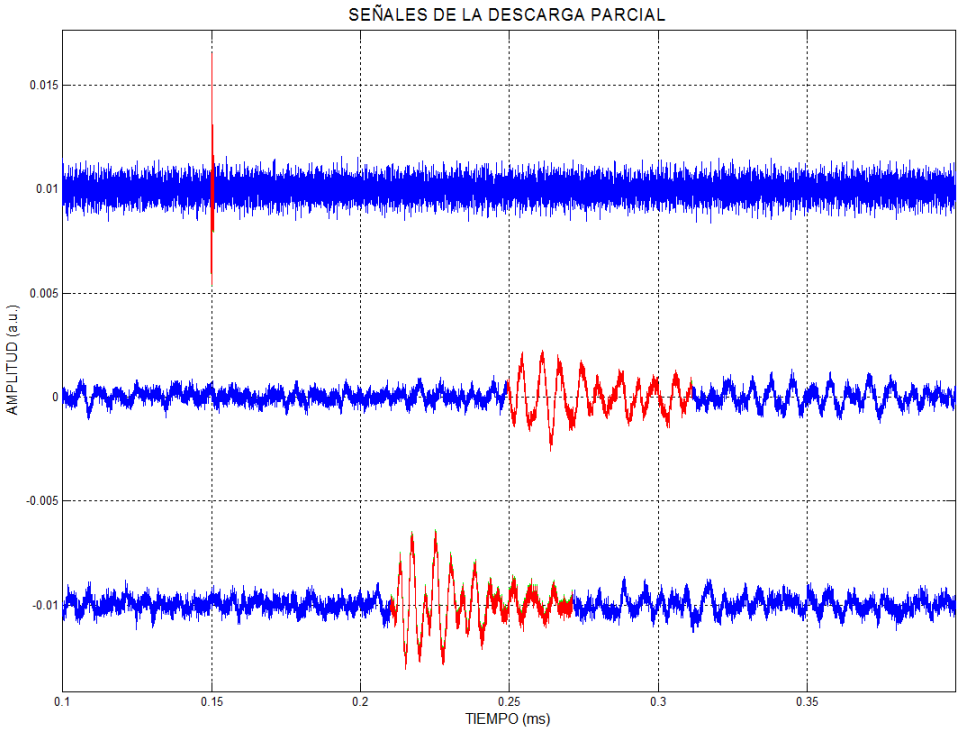


Figura 7.- Señal escogida como patrón en una de las adquisiciones [3]

A partir de un evento patrón como el de la Figura 7, se puede extraer las características de duración, forma de onda y retardo que se pueden utilizar para detectar señales con peor relación señal ruido. Una vez seleccionados los patrones de forma de onda de cada canal, se procede a filtrar la adquisición de cada canal para eliminar ruidos y acondicionar la señal.

Después de acondicionar los patrones y las adquisiciones, se detectan las señales de cada canal. Las señales encontradas en cada canal acústico se analizan a través de sus parámetros característicos y se almacenan.

Para detectar señales, se utiliza una herramienta de reconocimiento de patrones, en la cual, se escoge un patrón que se irá comparando con el resto de la señal mediante una correlación cruzada. Con esta técnica se detectan ventanas temporales en las señales, con una alta probabilidad de haber sido generadas por una DP.

2.2.2 Identificación

Una vez que se posee una base de datos de las señales encontradas en los canales acústicos, se procede a identificar cada una de las señales. Una señal detectada puede ser válida o no en función de unas condiciones temporales que deben cumplir las señales detectadas que componen un evento como se ha observado en la Figura 7.

A partir del parecido entre el patrón y las señales detectadas, y a partir de la relación temporal de las señales detectadas en los distintos canales, se identifica si los posibles conjuntos se corresponden a una DP.

2.2.3 Procesamiento de la señal eléctrica

Como la señal eléctrica se detecta cuando se produce la DP y, por tanto, se utiliza como referencia de tiempo cero para las señales acústicas detectadas, la detección de la señal eléctrica correspondiente a una DP se realiza en la ventana temporal correspondiente al inicio de la señal acústica y se retrocede en el tiempo hasta el valor de tiempo correspondiente a la mayor distancia que puede recorrer la señal acústica dentro de la cuba.

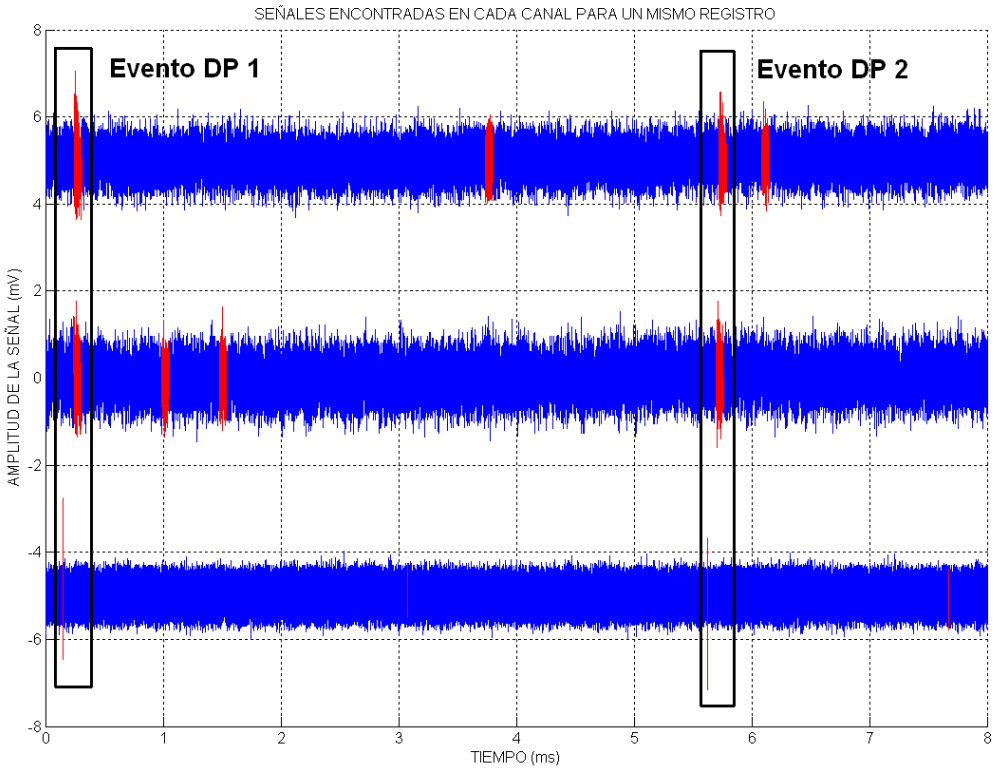


Figura 8.- Eventos DP válidos, dentro del recuadro [3]

En la Figura 8 se muestra el resultado del procesamiento en uno de los registros. Las señales encontradas en cada canal se hayan resaltadas en rojo. Las señales enmarcadas se corresponden a un posible evento DP; esto es, las dos señales acústicas y la señal eléctrica enmarcadas cumplen las condiciones temporales y de parecido necesarias para poder ser asociadas a un mismo evento.

A partir del retardo entre las señales acústicas y la eléctrica, se procede a encontrar el origen de la emisión acústica de la DP.

2.2.4 Localización

El usuario experto utiliza dos formas de localizar: la primera es la triangulación una vez detectada un grupo de DP en todos los canales (Figura 9) y la otra forma consiste en estudiar la actividad de cada sensor acústico por separado (Figura 10).

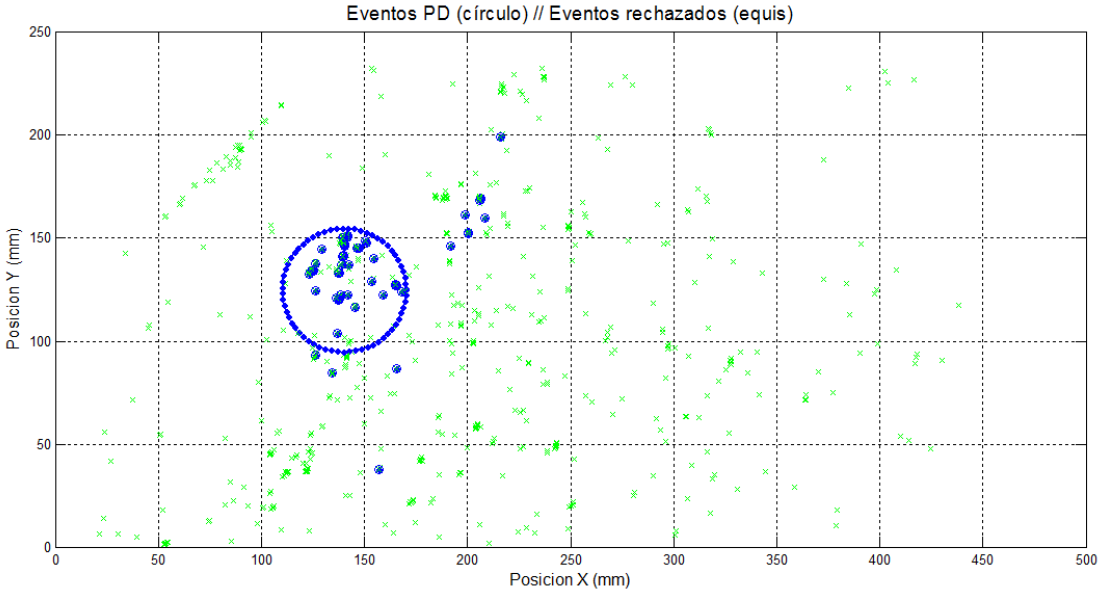


Figura 9.- Región seleccionada tras la localización por triangulación de eventos PD

La localización se basa en ubicar las regiones del espacio en torno a los sensores donde se concentra la mayor actividad de DP, a partir de la diferencia de tiempos entre la señal eléctrica (tomada como referencia de tiempo cero) y la señal acústica detectada por cada sensor y de la ubicación de los sensores acústicos. En la Figura 10 se muestra un resultado obtenido en el experimento de la Figura 4. Dentro del círculo amarillo es la zona donde habría más posibilidades de que se haya dado la DP. Para afirmar esto se tiene que comprobar mediante el análisis estadístico descrito en el siguiente apartado (2.2.5).

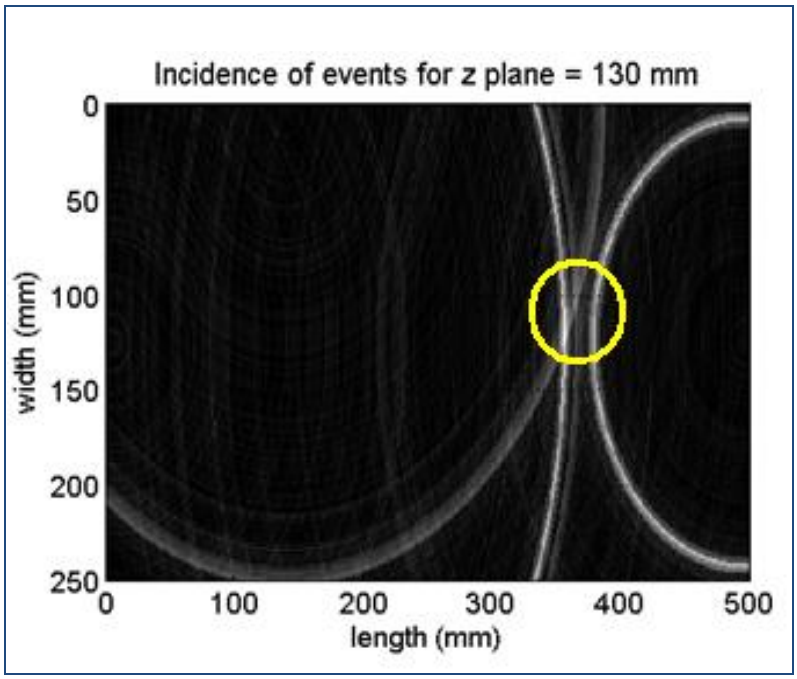


Figura 10.- Localización espacial con tres sensores

2.2.5 Análisis estadístico

Una vez parametrizados los eventos y localizados espacialmente, se realiza un análisis estadístico de las regiones con mayor incidencia de eventos y se agrupan aquellos que se concentran en una zona delimitada.

Por último, se procesan los diferentes parámetros de los eventos que se producen en una misma región para buscar la relación acústica y electro-acústica entre las señales.

2.3 Funciones y datos suministrados

A continuación se procede a realizar una descripción de los tipos de bases de datos suministradas y generadas por las funciones, así como las funciones diseñadas para realizar la detección y localización de DP.

2.3.1 Datos suministrados

- **Señales adquiridas:** Archivo de tipo *cell* (Figura 11) que contiene la información sobre la adquisición de las señales. Su nombre es de tipo numérico, normalmente es un número de cuatro dígitos (*Ej: 1070.mat*) que se corresponde a una secuencia de medidas experimentales desde la primera adquisición 1001 hasta la 9999. Este formato permite abrir y procesar un conjunto grande de medidas de forma secuencial. En estos ficheros se guardan las variables: valor de DC (*DCchx*[1]), frecuencia de muestreo (*Fsx*), la señal (*chx*), el incremento temporal (*dtchx*) y estado de cada uno de los canales adquiridos (representados por el número del 1 al 4 al final de cada variable). Estos datos serán característicos de cada canal adquirido (si existen el canal 1 y el canal 3, existirán DCch1, DCch3, Fs1, Fs3, y así sucesivamente). Las señales suministradas contienen 4 canales, 3 de ellos de tipo acústico y otro de tipo eléctrico.

[1] La *x* representa el número del canal de la señal adquirida. Las señales cuentan con 4 canales, por lo que *x* variará entre 1 y 4.

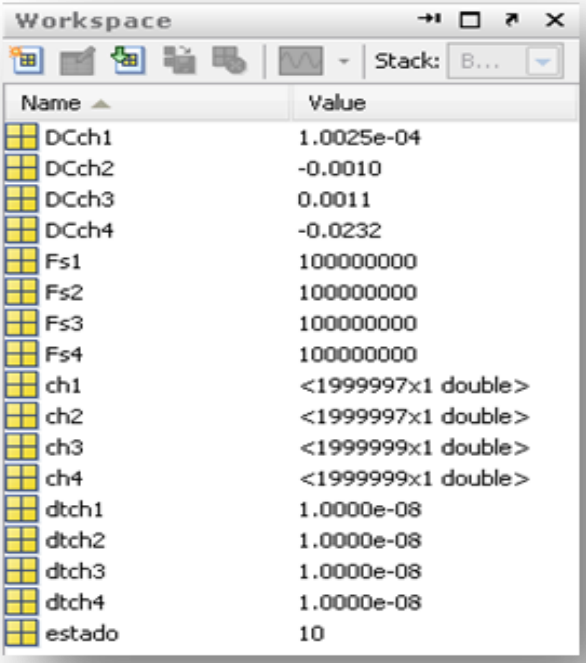


Figura 11.- Datos de una adquisición típica de 4 canales

Para representar las señales, los datos de interés son *chx* y *dtchx*. *Chx* contiene los valores instantáneos que tiene la señal en cada muestreo realizado en la adquisición. *Dtchx* es el incremento temporal que hay entre un valor y el siguiente de la señal. A la hora de reconstruirla se realiza el siguiente procedimiento: se sitúa cada muestra respetando el orden desde la primera a la última con una separación entre una y otra de *dtchx* segundos. En la Figura 12 se muestra una señal representada de la forma que se ha explicado.

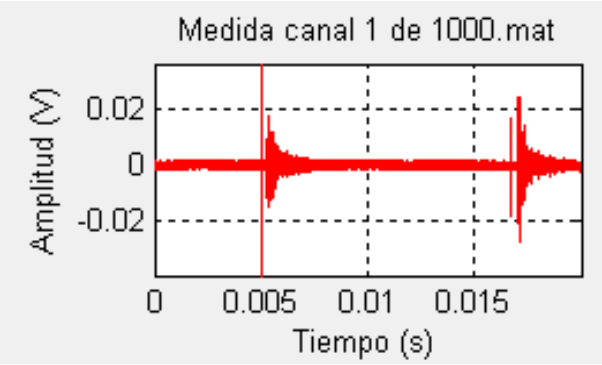


Figura 12.- Representación de un canal de la adquisición

- **Patrón:** Archivo de tipo *cell* (Figura 13) que contiene la información sobre el patrón creado. Del mismo modo que el archivo de señal, su nombre es de tipo numérico, en este caso de 7 dígitos. Los 4 primeros representan la señal de origen del patrón y los 3 últimos representan el canal del que se ha obtenido dicho patrón (*Ej: 1070002.mat* corresponde al patrón del canal 2 extraído de la señal 1070.mat). En este tipo de archivo se guardan las siguientes variables: *Fcentrales,*

Patrondata, desc_level, envpatronw, orderwpatron, patron, patronw, waveletname, wdproppatron, wdproppatron.TXT.

Para generar estos datos hay que seleccionar una ventana temporal dentro de una señal adquirida, que se considere emitida por una DP, esta ventana temporal seleccionada es la forma de onda patrón para el canal del que ha sido extraída. El patrón lo selecciona el experto a partir de la experiencia de observación de las señales experimentales, la ventana temporal asociada se almacena en una variable que se llama *patron* dentro de un archivo *patronx.mat*. Debe ser así ya que la función está diseñada exclusivamente para este procedimiento. Solo tras crear este archivo se podrá ejecutar la función *patronizar* o *patronizarw* que se describirá en el apartado dedicado al estudio de las funciones suministradas (2.3.2). Ambas funciones crean el archivo de tipo patrón.

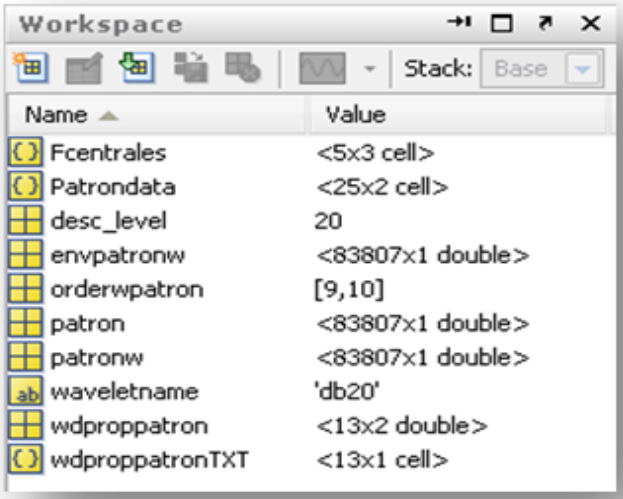
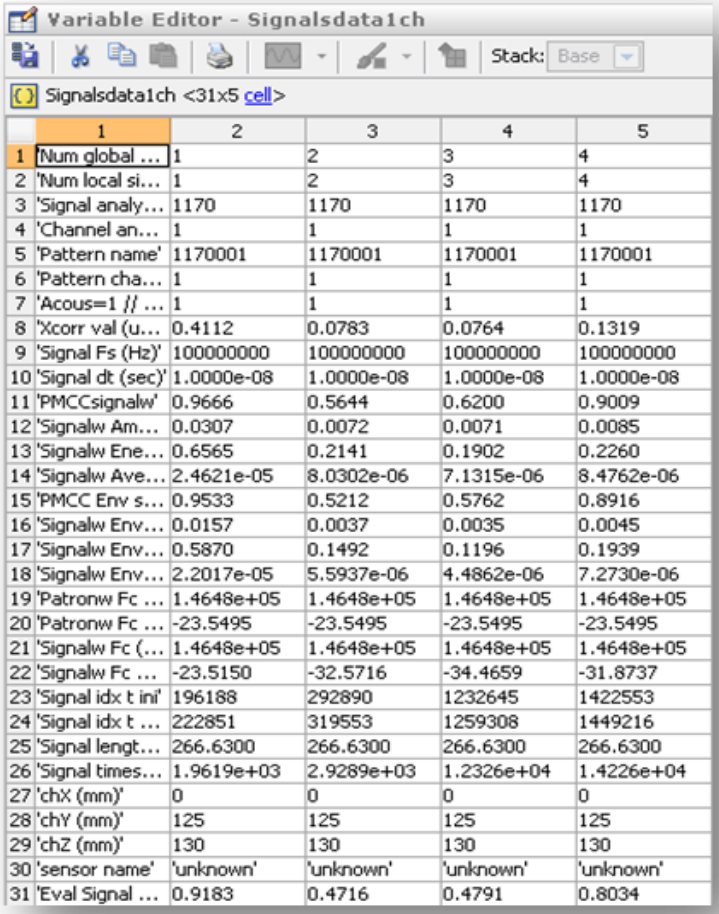


Figura 13.- Datos característicos del patrón

- **Registro de señales encontradas:** Archivo de tipo *cell* (Figura 14) que contiene la información sobre la búsqueda de posibles señales DP acústicas usando patrones en un rango de señales. En este tipo de archivo (nombrado generalmente como *reg100x.mat*) se guardan los parámetros de las señales acústicas encontradas y ciertos valores de interés en la variable *Signalsdata1ch*, así como los datos de la búsqueda de la que provienen esos datos en *regsearch*, como pueden ser rango de señales, patrón empleado o número de canal. *Signalfinder1ch* será la función que proporcione este tipo de archivos.



Variable Editor - Signalsdata1ch

Stack: Base

Signalsdata1ch <31x5 cell>

	1	2	3	4	5
1	'Num global ...	1	2	3	4
2	'Num local si...	1	2	3	4
3	'Signal analy...	1170	1170	1170	1170
4	'Channel an...	1	1	1	1
5	'Pattern name'	1170001	1170001	1170001	1170001
6	'Pattern cha...	1	1	1	1
7	'Acous=1 // ...	1	1	1	1
8	'Xcorr val (u...	0.4112	0.0783	0.0764	0.1319
9	'Signal Fs (Hz)'	100000000	100000000	100000000	100000000
10	'Signal dt (sec)'	1.0000e-08	1.0000e-08	1.0000e-08	1.0000e-08
11	'PMCCsignalw'	0.9666	0.5644	0.6200	0.9009
12	'Signalw Am...	0.0307	0.0072	0.0071	0.0085
13	'Signalw Ene...	0.6565	0.2141	0.1902	0.2260
14	'Signalw Ave...	2.4621e-05	8.0302e-06	7.1315e-06	8.4762e-06
15	'PMCC Env s...	0.9533	0.5212	0.5762	0.8916
16	'Signalw Env...	0.0157	0.0037	0.0035	0.0045
17	'Signalw Env...	0.5870	0.1492	0.1196	0.1939
18	'Signalw Env...	2.2017e-05	5.5937e-06	4.4862e-06	7.2730e-06
19	'Patronw Fc ...	1.4648e+05	1.4648e+05	1.4648e+05	1.4648e+05
20	'Patronw Fc ...	-23.5495	-23.5495	-23.5495	-23.5495
21	'Signalw Fc (...	1.4648e+05	1.4648e+05	1.4648e+05	1.4648e+05
22	'Signalw Fc ...	-23.5150	-32.5716	-34.4659	-31.8737
23	'Signal idx t ini'	196188	292890	1232645	1422553
24	'Signal idx t ...	222851	319553	1259308	1449216
25	'Signal lengt...	266.6300	266.6300	266.6300	266.6300
26	'Signal times...	1.9619e+03	2.9289e+03	1.2326e+04	1.4226e+04
27	'chX (mm)'	0	0	0	0
28	'chY (mm)'	125	125	125	125
29	'chZ (mm)'	130	130	130	130
30	'sensor name'	'unknown'	'unknown'	'unknown'	'unknown'
31	'Eval Signal ...	0.9183	0.4716	0.4791	0.8034

Figura 14.- Señales encontradas tras la detección, almacenadas en el archivo reg100x.mat

- **Registro de señales DP encontradas:** Archivo de tipo *cell* que contiene la información sobre las DP encontradas. En este tipo de archivo (nombrado generalmente como *regPDdatax.mat*) se guardan las señales acústicas encontradas asociadas a un evento eléctrico y sus datos característicos en la variable *PDdata.* Este tipo de datos se obtiene aplicando las funciones *Globalacusfinder* y las funciones *acuselecaso* para cada canal.

- **Registro de señales acústicas asociadas:** Archivo de tipo *cell* que contiene la información sobre las señales acústicas asociadas a otras señales acústicas que cumplen las condiciones temporales para considerarse ambas asociadas a un mismo evento. En este tipo de archivo (nombrado generalmente como *regacusmix.mat*) se guardan las señales acústicas asociadas y sus datos característicos en la variable *Acusmixedsignals.* Se obtiene del uso combinado de las funciones *Globalacusfinder* y *AcusPDmixingv2.*

2.3.2 Funciones suministradas

Se realiza una breve descripción de las funciones principales utilizadas para las distintas etapas de la detección y localización las descargas parciales.

- **Signalfinder1signal:** Busca posibles señales DP en una única señal (*filenamenum*), con un patrón (*patronnum*) y en un único canal (*chsignal*). Esta función compara el patrón con la señal y devuelve información sobre los fragmentos de la señal que considera que se parecen al patrón. Los argumentos de entrada de esta función son el nombre de una señal, el nombre de un patrón y el canal a analizar. Como salida creará la variable *Signalsdata* que contiene valores característicos de las señales encontradas que se han calculado mediante las distintas funciones de cálculo (Figura 15).

  [Signalsdata] = Signalfinder1signal (filenamenum, patronnum, chsignal)

Figura 15.- Esquema de la función Signalfinder1signal.m

- **Signalfinder1ch:** Busca posibles señales DP en un único canal para un conjunto de adquisiciones. Esta función (Figura 16) emplea *Signalfinder1signal* para analizar un intervalo de señales del mismo canal con el mismo patrón. Para ello comprueba que existe el canal en esa señal. Si es así, analiza con *Signalfinder1signal* el canal de dicha señal. Si no encuentra el canal, pasa a la siguiente señal y realiza la misma tarea. Los argumentos de entrada de esta función son la primera adquisición del intervalo de medidas del laboratorio (*filenameini*), la última adquisición del intervalo (*filenameend*), el nombre del patrón (*patronnum*) que se va a utilizar y el canal a analizar (*chsignal*).

  Como salida se genera la variable *Signaldata1ch*, que es un conjunto de las *Signalsdata* generadas por las múltiples llamadas que se hacen a la función *Signalfinder1signal*. Estas contienen la información de las señales que se han encontrado.

  [Signalsdata1ch, filenamesave] = Signalfinder1ch (filenameini, filenameend, patronnum, chsignal)

Figura 16.- Esquema de la función Signalfinder1ch.m

- **Globalacusfinder:** Busca posibles señales DP en un intervalo de adquisiciones analizando todos sus canales acústicos (Figura 17). Los argumentos de entrada de esta función son la primera adquisición del intervalo de medidas (*filenameini*), la última adquisición del intervalo (*filenameend*) y el nombre de la señal que contiene el patrón (*filecontainpatron*). La función únicamente analiza las adquisiciones y canales existentes, que además tengan un patrón definido.

  Esta función al final almacena en un archivo (generalmente nombrado *Regacusfile.mat*) los nombres de los ficheros procesados, y también crea los registros de las señales encontradas (*reg100x.mat*) donde se almacenan los datos de las señales encontradas.

  [Regacus, numreg] = Globalacusfinder (filenameini, filenameend, filecontainpatron)

Figura 17.- Esquema de la función *Globalfinder.m*

- **Acuselecasoc1, 2 y 3:** Estas funciones (Figura 18) sirven para asociar una señal eléctrica a un evento acústico . Se utiliza tras haber buscado posibles señales DP en todos los canales con *Globalacusfinder*. Los argumentos de entrada son el registro de señales encontradas (*registerfile*) generado en cada canal por la función *Globalacusfinder*, el nombre del patrón eléctrico (*patronnumelec*) y el canal donde se encuentra este patrón eléctrico (*chsignalelec*). Como salida creará el registro de señales DP encontradas (*regPDdatax.mat).*

  acuselecasoc1 (registerfile, patronnumelec, chsignalelec)

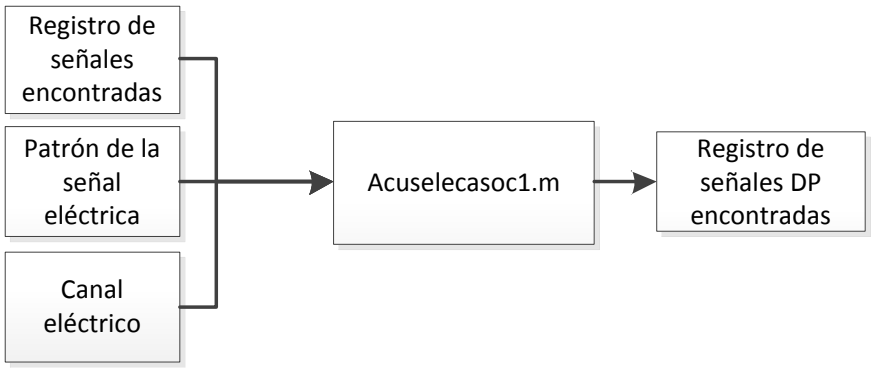


Figura 18.- Esquema de la función Acuselecasoc1.m

- **AcusPDmixingv2:** Esta función se emplea en la localización por triangulación donde es necesario que las señales acústicas detectadas en los distintos canales estén asociadas a un mismo evento DP. Para ello, busca en los registros generados con *Globalacusfinder* y trata de emparejar por tiempos las distintas señales acústicas (Figura 19). Emparejar por tiempos significa que busca los posibles emparejamientos tales que la diferencia entre los tiempos máximo y mínimo hallados se encuentre en un rango entre 0 y el tiempo que tardaría en recorrer la distancia entre sensores.

  [Acusmixedsignals] = AcusPDmixingv2

Figura 19.- Esquema de la función AcusPDmixingv2.m

- **Patronizar:** Esta función (Figura 20) genera archivos tipo patrón *(Ej: 1170003.mat)* a partir de una selección de un evento, escogida por un experto, que se considere válido como patrón de descarga parcial en una de las señales adquiridas. Esta selección se guarda en una variable llamada *patron*. Los argumentos de entrada de esta función son el nombre de la señal original de la que se ha seleccionado el evento (*filenamepatnum*), el canal del cual se ha adquirido el evento (*chpat*) y la naturaleza de la señal (*acus_o_elec*) ya sea acústica o eléctrica. Se recomienda crear patrones para todos los canales, ya que de no ser así

  patronizar (filenamepatnum, chpat, acus_o_elec)

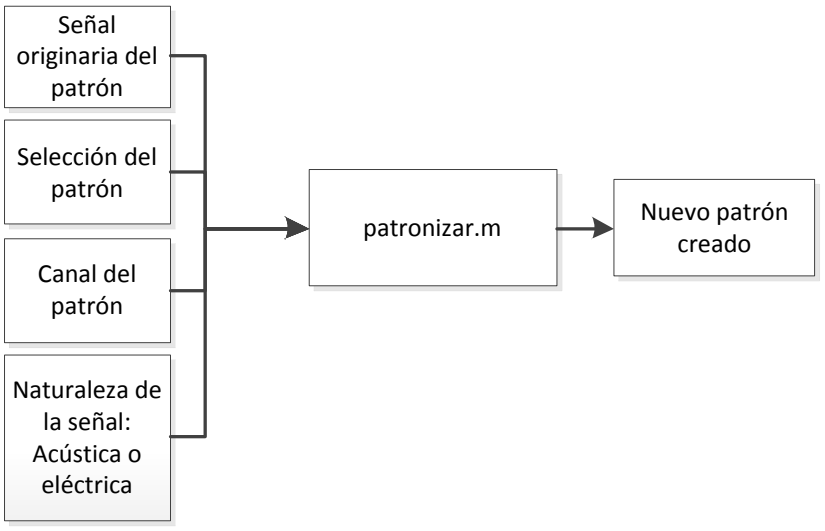


Figura 20.- Esquema de la función *patronizar.m*

- **Patronizarw:** Esta función tiene la misma utilidad que la función *patronizar.m*, únicamente que la selección del evento pasa a llamarse *patronw*, y generará un archivo de tipo patrón, realizando un filtrado wavelet a la señal. Los argumentos (Figura 21) de entrada serán los mismos que en la función anterior.

  patronizarw (filenamepatnum, chpat, acus_o_elec)

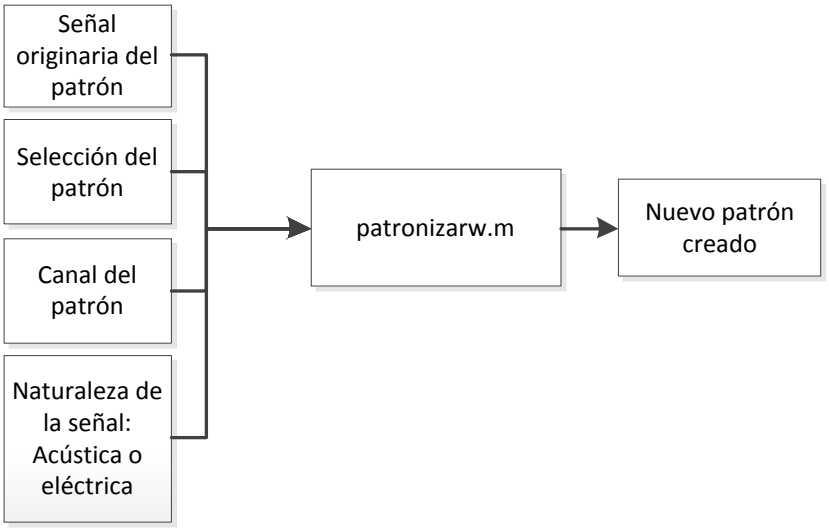


Figura 21.- Esquema de la función *patronizarw.m*

- **PlotSignalFind:** Pinta las señales encontradas en los canales cuando se ha realizado una búsqueda que ha generado registros de señales encontradas (*reg100x.mat*). Los argumentos (Figura 22) de entrada son el nombre de la señal (*filenamenum*) de la que se quiere ver las señales encontradas y el registro de señales encontradas correspondientes al canal deseado (*reg100x.mat*).

  PlotSignalFind (filenamenum, reg100x)

Figura 22.- Esquema de la función *PlotSignalFind.m*

- **pintaPD:** Pinta las DP detectadas sobre las señales originales (Figura 23) tras realizar una búsqueda con *Globalacusfinder* y ejecutar las funciones *acuselecasoc1, 2 y 3*. Se deben generar los registros de señales DP encontradas para todos los canales existentes, sin ellos la función dará error. El argumento de entradas el nombre de la señal de la que se desean pintar las DP encontradas.

  pintaPD (filenamenum)

Figura 23.- Esquema de la función *PintaPD.m*

- **pintapatron:** Pinta el patrón seleccionado sobre su señal original (Figura 24). El argumento de entrada es el nombre de la señal de la que se ha adquirido el patrón.

  pintapatron (filenamenum)

Figura 24.- Esquema de la función *pintapatron.m*

- **pinta4ch:** Reconstruye y pinta las señales adquiridas. Representa en una ventana las 4 canales juntas y en ventanas individuales los distintos canales (Figura 25). El argumento de entrada es el nombre de la señal que se desea pintar (*filename*).

  pintapatron (filenamenum)

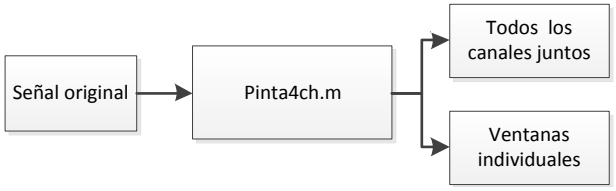


Figura 25.- Esquema de la función pinta4ch

- **Lanzador2D:** Realiza la localización espacial de los eventos encontrados (Figura 26). Pinta la lectura de los 3 canales acústicos juntos utilizando la función *Histograma2D* que proporciona información sobre las señales encontradas. Los argumentos de entrada son todos aspectos de la representación gráfica como puede ser la posición que queremos representar o la elección de una representación individual de cada uno de los análisis. Es necesario que existan los registros de las señales DP encontradas, ya que sin estos registros la función no se ejecutará correctamente.

  Lanzador2D(pintar,pintaindividual,pinta_tiempo,pintar3D,factorsub,minfactor,vsound_min,vsound_max,vsound_paso,zmedida_min,zmedida_max,zmedida_paso)

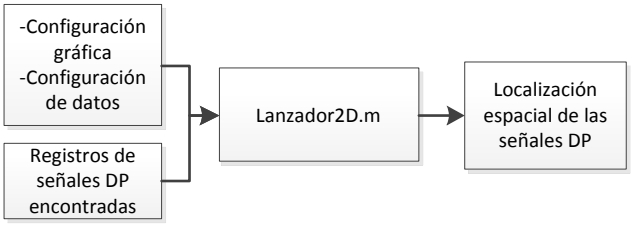


Figura 26.- Esquema de la función Lanzador2D.m

- **Histograma2D:** Representa mediante histogramas las descargas parciales en función de la distancia al sensor o del tiempo que tarda la DP en llegar al sensor (Figura 27). Los argumentos de entradas son el nombre de la señal que se desea analizar *(filename)*, la velocidad de transmisión del sonido en el medio en el que se trabaja (*vsound*), la altura de la cuba a la que se realiza el análisis (*zmedida*) y un valor que permite o impide que se pinte la señal (*pintar*).

  Histograma2D (filename, vsound, zmedida, pintar)

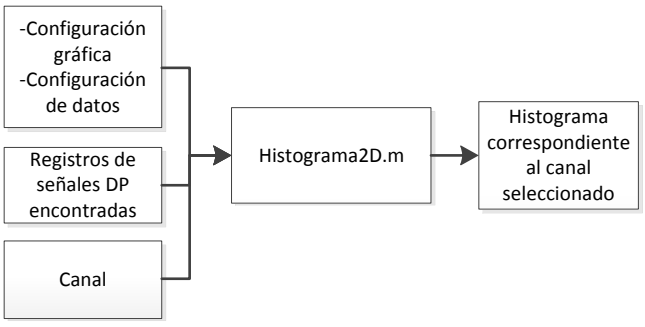


Figura 27.- Esquema de la función Histograma2D.m

Existen más funciones que no se describen debido a que tras el análisis se ha llegado a la conclusión de que su utilidad es de configuración de ciertos aspectos matemáticos que no entran dentro de la utilidad de la herramienta sino en los cálculos necesarios para obtener cierta información para el procesado.

2.3.3 Conclusiones

Analizando las funciones suministradas y la bibliografía [2], [3] se diferencian los distintos escenarios de trabajo.

2.3.3.1 Creación de un patrón

A continuación se describe el proceso que se debe seguir para crear un patrón utilizando las funciones dadas.

Como se ha visto anteriormente se debe seleccionar un evento válido de una señal y guardarlo en una variable con el nombre "patron" dentro de un archivo con el nombre *patronX.mat*, siendo la *X* el canal del que se ha extraído el patrón.

Una vez almacenada esta selección utilizaremos la función *patronizar.m* para realizar un filtrado de la señal y un primer procesado de esta. Tras realizar el primer procesado del patrón, ejecutaremos la función *patronizarw.m* con los datos generados por *patronizar.m*. Este proceso crea un patrón calculando todos sus valores característicos necesarios para utilizarlo en funciones de detección.

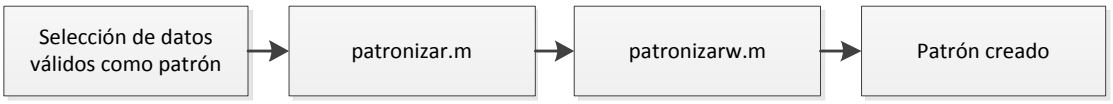


Figura 28.- Proceso de creación de patrón

2.3.3.2 Detección de señales DP

A continuación se describen los distintos tipos de búsqueda que se pueden realizar utilizando las funciones suministradas.

- **<u>Búsqueda en un solo canal:</u>** La función asociada a este tipo de búsqueda es *Signafinder1ch.m* (Figura 29). Este tipo de búsqueda analiza un único canal de una señal con un patrón para encontrar señales en ella.

Figura 29.- Esquema de la búsqueda en un solo canal

- **Búsqueda global acústica:** Este tipo de búsqueda analiza un intervalo de señales con un patrón y asocia entre sí las distintas señales acústicas encontradas. Las funciones asociadas a este tipo de búsqueda son *Globalacusfinder.m y AcusPDmixingv2.m* (Figura 30).

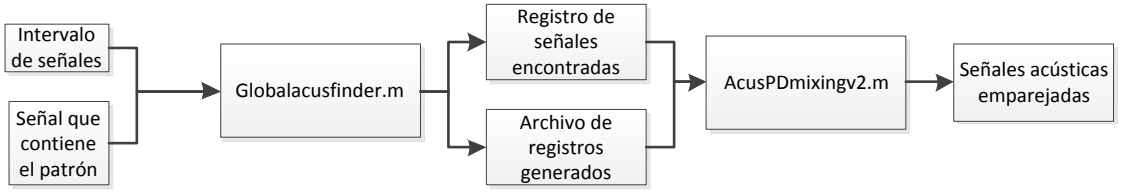


Figura 30.- Esquema de la búsqueda global acústica

- **Búsqueda con referencia temporal:** Este tipo de búsqueda analiza un intervalo de señales con un patrón y asocia las señales encontradas con algún evento eléctrico. Las funciones asociadas a este tipo de búsqueda son *Globalacusfinder.m, más Acuselecasoc1.m, 2.m y 3.m y más DelayCalc.m* (Figura 31).

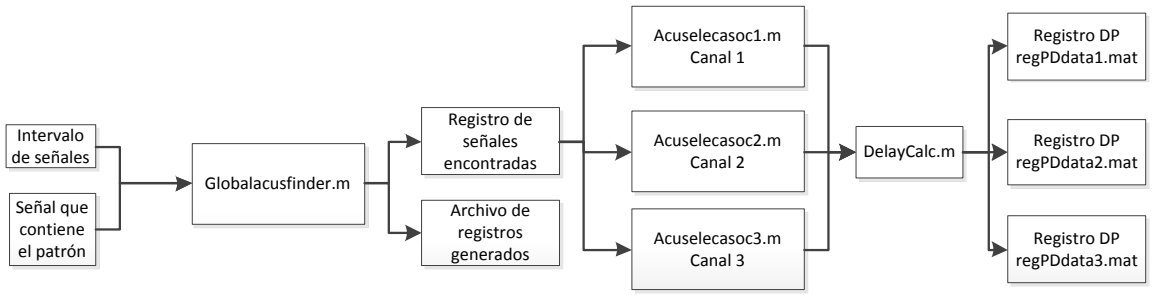


Figura 31.- Esquema de la búsqueda con referencia temporal

2.3.3.3 Visualización de las señales encontradas

Existen dos tipos fundamentales de visualización de las señales encontradas en función de la búsqueda realizada. El primero es utilizando la función *PlotSignalFind.m.* Esta función se podrá utilizar tras realizar una búsqueda de tipo global acústico o en un único canal (Figura 32).

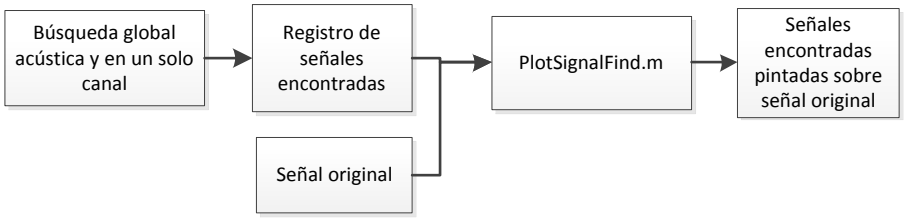


Figura 32.- Esquema de visualización para búsqueda global acústica o para un solo canal

Existe otro tipo de tipo de visualización que es la realizada tras una búsqueda con referencia temporal. Esta permite mostrar las descargas parciales de un mismo evento

viendo los 4 canales en la misma ventana. El esquema de funcionamiento es el descrito en la Figura 33.

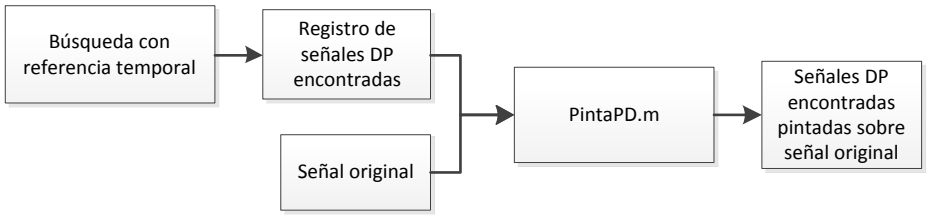


Figura 33.- Esquema de visualización para búsqueda global con referencia temporal

2.3.3.4 Localización espacial de señales

La localización espacial por actividad (que es la que el experto utiliza) se basa en la función *Lanzador2D.m*. Para ello se configuran ciertos aspectos de la representación que permite la función. Estos son principalmente el tipo de representación que en este caso se divide en dos. El primero es una localización por planos (Figura 34) en la que se representará un corte de la cuba a una altura *Z* en la que se muestran con circunferencias las distancias a las que los sensores han detectado las DP.

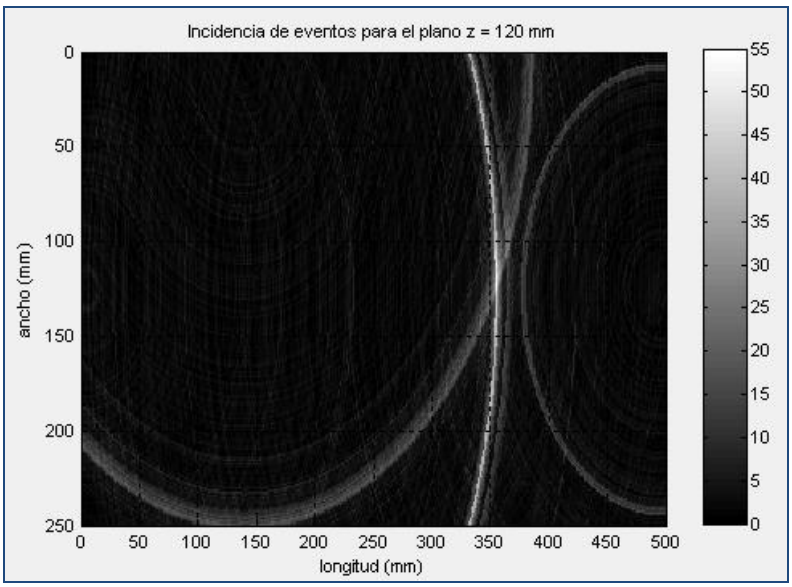


Figura 34.- Representación por planos con la función *Lanzador2D*

El segundo tipo es la representación tridimensional, muy parecida a la localización por planos. La representación tridimensional muestra en tres dimensiones las zonas con mayor o menor número de eventos detectados (Figura 35) en un plano *Z* de la cuba.

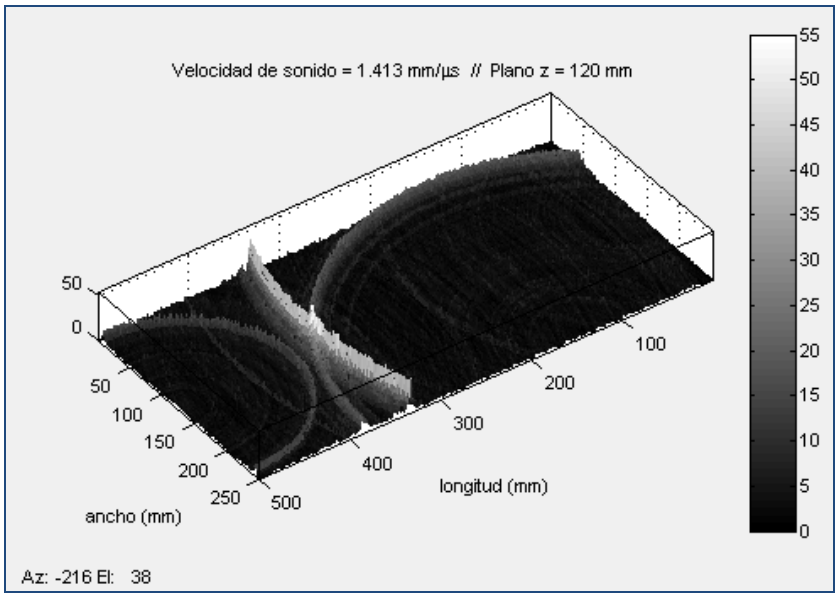


Figura 35.- Representación tridimensional la función *Lanzador2D*

Se puede mostrar más información relativa a la localización espacial mediante la función *Histograma2D*. Esta función muestra mediante histogramas la distancia a la que se ha dado cada evento o el tiempo que ha tardado en llegar al sensor (Figura 36).

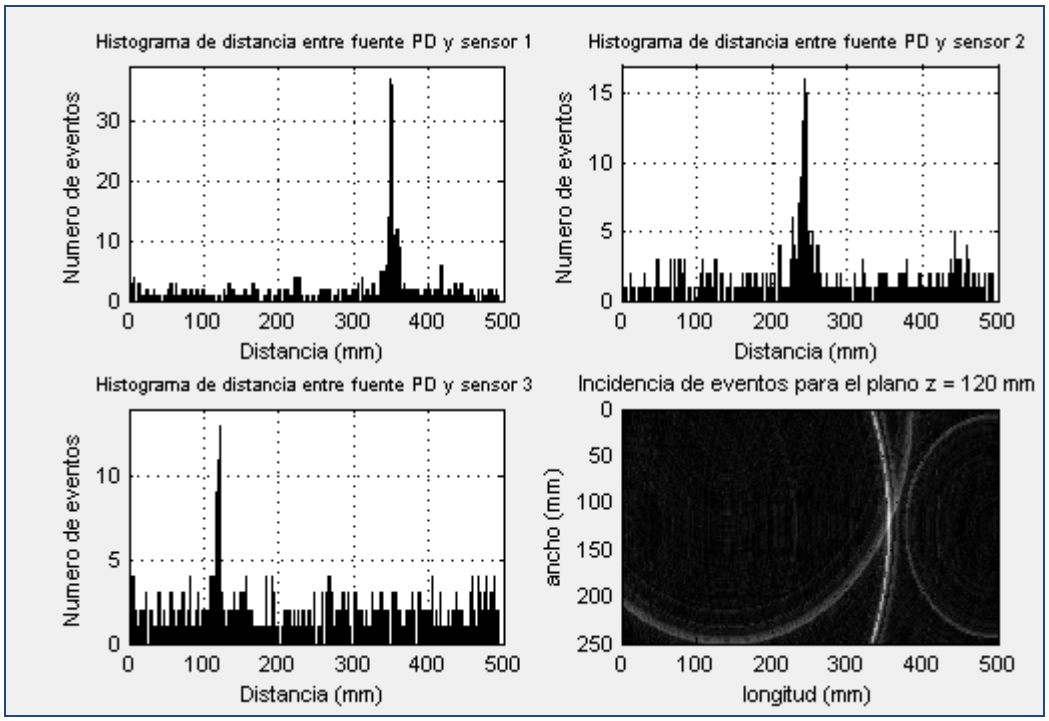


Figura 36.- Representación de histogramas la función *Histograma2D*

2.3.3.5 Esquema secuencial de las funciones

Del análisis realizado en este capítulo, y basado en las conclusiones en las que se definen los distintos escenarios de trabajo, se puede esbozar un esquema (Figura 37) del funcionamiento total del conjunto de funciones suministradas y del flujo de datos necesario para su correcto funcionamiento. Representa la totalidad de las funciones trabajando de forma secuencial, desde los primeros pasos como son el crear un patrón hasta la detección de DP o la localización espacial de estas.

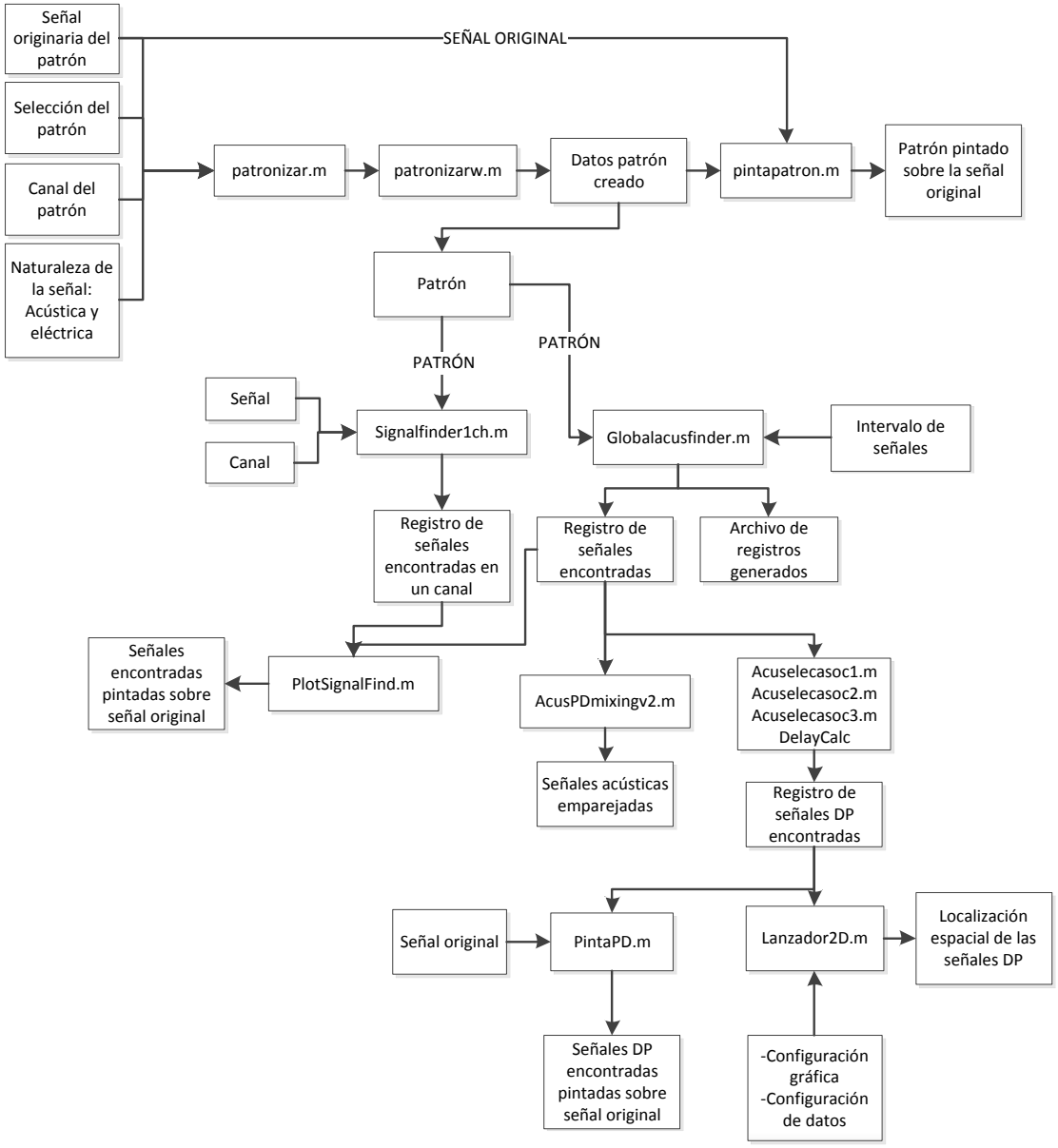


Figura 37.- Esquema de la detección y localización de descargas parciales secuenciando todas las funciones

Capítulo 3

Herramienta de desarrollo y diseño de una interfaz

3.1 Interfaces de usuario con MATLAB

3.1.1 MATLAB

MATLAB © (abreviatura de *MATrix LABoratory*, "laboratorio de matrices"), Figura 38, es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X [4].

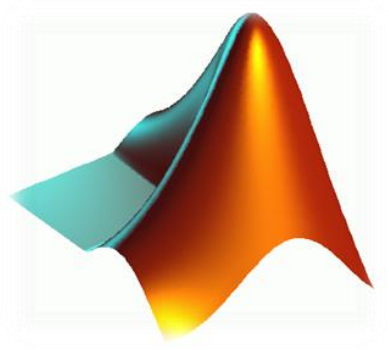


Figura 38.- Logo MATLAB

Fue creado por *Cleve Moler* en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices *LINPACK* y *EISPACK* sin tener que usar Fortran.

Es un programa de cálculo numérico, orientado a matrices y vectores. Por tanto desde el principio hay que pensar que todo lo que se pretenda hacer con él será mucho más rápido y efectivo si se piensa en términos de matrices y vectores. Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (*toolboxes*).

MATLAB es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

3.1.2 GUIDE en MATLAB

El presente proyecto se centra en una de las utilidades más valiosas del paquete MATLAB, como es la creación de interfaces de usuario con un gran abanico de posibilidades y gran sencillez para el usuario de las mismas una vez diseñadas mediante MATLAB GUIDE [4][5][6][7].

GUIDE es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo de datos. Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++, pero haciendo más intuitiva e inmediata la programación y el acceso a las diferentes funcionalidades que ofrece.

La interfaz gráfica de usuario, conocida también como GUI (del inglés *graphical user interface*) es un programa informático que actúa de canal de comunicación entre humano y ordenador, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la interacción con el sistema operativo de una máquina o con un ordenador sin la necesidad de poseer grandes conocimientos al respecto.

Habitualmente las acciones se realizan mediante manipulación directa. Surge como evolución de los intérpretes de comandos que se usaban para operar los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. Como ejemplo de interfaz gráfica de usuario cabe citar los entornos de escritorio de Windows.

Dentro de las Interfaces de Usuario se puede distinguir básicamente tres tipos:

- *Una interfaz de hardware*, a nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla.

- *Una interfaz de software*, destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observa habitualmente en la pantalla

- *Una interfaz de Software-Hardware*, que establece un puente entre la máquina y las personas, permite a la maquina entender la instrucción y a el hombre entender el código binario traducido a información legible.

Atendiendo a como el usuario puede interactuar con una interfaz, se dan varios tipos de interfaces de usuario:

- *Interfaces alfanuméricas* (intérpretes de mandatos) que solo presentan texto.

- *Interfaces gráficas de usuario* (GUI), las que permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.

- *Interfaces táctiles*, que representan gráficamente un "panel de control" en una pantalla sensible que permite interaccionar con el dedo de forma similar a si se accionara un control físico.

Este proyecto se centra en el diseño y desarrollo de una interfaz gráfica de usuario ya que el intercambio de información entre el usuario y el ordenador se debe comprender gráficamente, mostrando datos y gráficas por medio de los distintos elementos de control.

3.1.3 Programación con GUIDE

Para comprender el funcionamiento de las GUI en MATLAB se debe tener en cuenta que la programación se divide en dos partes fundamentales: una gráfica y otra de código de texto.

Por lo tanto una aplicación GUIDE consta de dos archivos: *.m* y *.fig*. El archivo *.m* es el que contiene el código con las correspondencias de los botones de control de la interfaz y el archivo *.fig* contiene los elementos gráficos.

- **Fichero *.fig*:**

Es el fichero que comprende la parte visual de la interfaz de usuario. En él se registran todos los componentes que conforman la aplicación, así como sus propiedades y algo muy importante, como es el *Tag* o etiqueta de cada elemento de la GUI, mediante el cual se podrá identificar el mismo para la posterior programación en el archivo .m. como se muestra en el apartado de Programación de la interfaz (3.3)

Se podrán modificar los contenidos de este fichero en cualquier con el editor, como se explicará en el apartado siguiente (3.2).

- **Fichero *.m*:**

Es un fichero en formato de texto que tiene una parte generada por el programa, con contenidos tales como las funciones de inicialización, los *callbacks* (se explicarán en el apartado de programación de la interfaz (3.3)) y las funciones para cada elemento de la

GUI, con el objetivo de que el diseñador pueda programar cada uno de los elementos. Cada vez que se añada un nuevo elemento en la interfaz gráfica, se genera automáticamente su código correspondiente en el archivo *.m*.

3.2 Generación del fichero gráfico de la interfaz

A continuación se pasará a exponer el método empleado para tratar y generar el archivo *.fig* (por consiguiente se tendrá también el archivo *.m* que se explicará en el siguiente apartado), así como unas indicaciones básicas acerca de su utilización.

Se accede al GUIDE una vez ejecutado MATLAB, y se puede hacer de dos formas distintas:

- Escribiendo la instrucción >> *guide* en la ventana de comandos.
- Haciendo clic en el icono correspondiente a GUIDE (Figura 39).

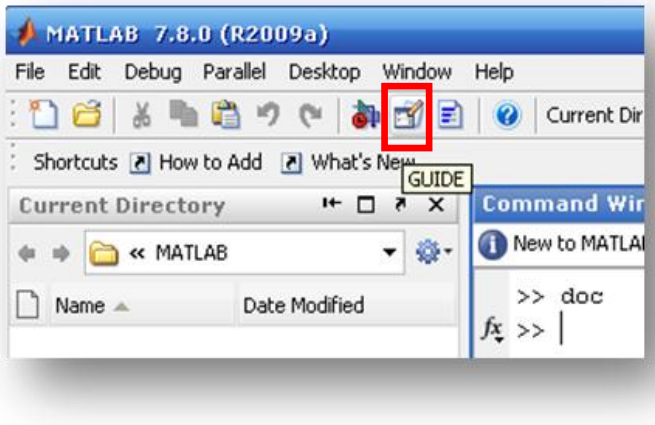


Figura 39.- Icono de acceso a la herramienta GUIDE

Una vez ejecutada una de las dos opciones se presentará el cuadro de diálogo representado en la Figura 40. Dentro de dicho cuadro de diálogo se presentan las siguientes cuatro opciones de partida:

- ***Blank GUI (Default)***

Se trata de la opción de interfaz gráfica de usuario en blanco predeterminada, por ser la más utilizada. Presenta un formulario nuevo en el cual se podrá diseñar una determinada interfaz desde la pantalla básica sin ningún botón predeterminado.

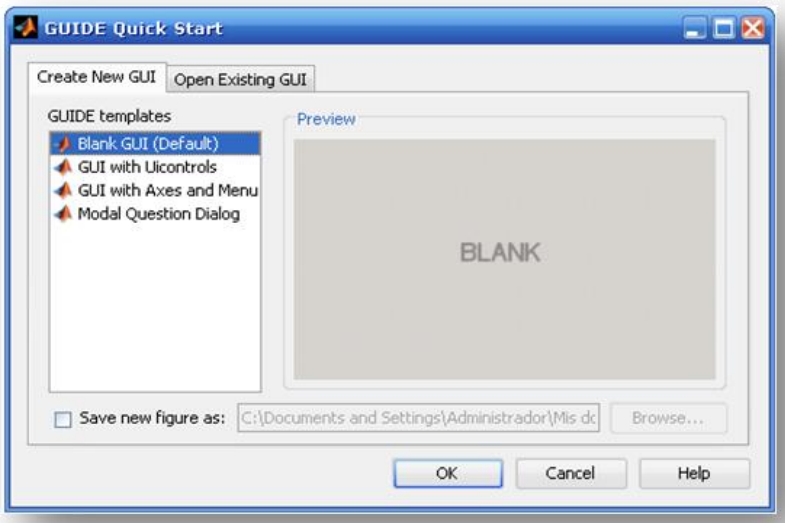


Figura 40.- Ventana de inicio de la herramienta GUIDE

- ***GUI with Uicontrols***

Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen del objeto, en cualquiera de los dos sistemas de unidades más extendidos. En esta opción se puede ejecutar el ejemplo y obtener resultados a partir del mismo.

- ***GUI with Axes and Menu***

Esta opción es otro ejemplo en el cual se incluye un menú *File* con las opciones *Open*, *Print* y *Close* en su interior. En el formulario cuenta con un *Popup menu*, un *Push button* y un objeto *Axes* (se explicarán estos objetos en el apartado 3.2.1). Se puede ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú desplegable, para posteriormente hacer clic en el botón de comando.

- ***Modal Question Dialog***

Con esta opción se muestra por pantalla un cuadro de diálogo común, que consta de una pequeña imagen, una etiqueta y dos botones, con las opciones *Yes* y *No* respectivamente. Dependiendo del botón que se presione, el GUI en cuestión devolverá el texto seleccionado (la cadena de caracteres *Yes* o *No*).

Tras seleccionar la opción *Blank GUI (Default)* se creará un entorno de diseño en blanco (Figura 41), pudiéndose diferenciar las herramientas que a continuación se detallan.

Los elementos básicos de la herramienta GUIDE son:

- **Componentes**

Seleccionando el componente dentro de este menú, solo hay que dimensionarlo en la ventana pinchando y arrastrando con el ratón. El funcionamiento de todos ellos está descrito en el apartado Componentes de las Guides (3.2.1).

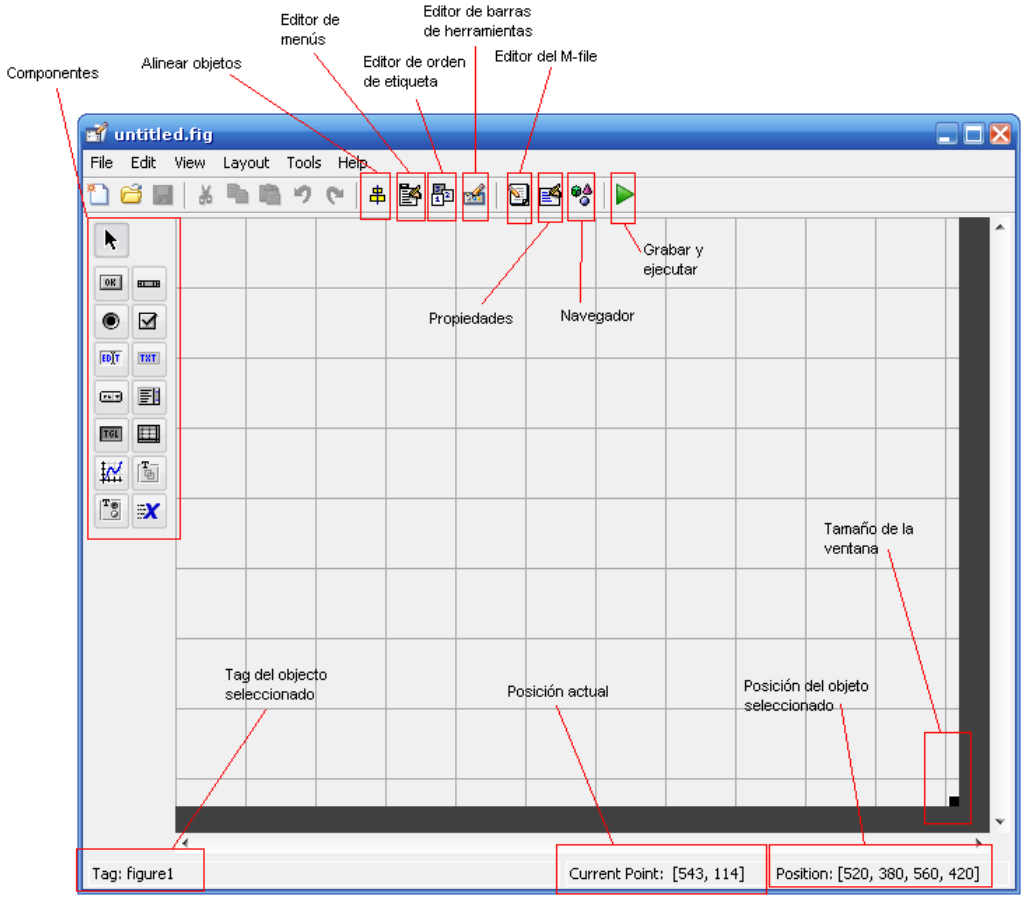


Figura 41.- Herramientas presentes en el entorno de diseño de la herramienta GUIDE

- **Posición actual**

Muestra la posición actual del cursor. Esta opción es útil para conocer puntos exactos de la interfaz, en caso de ser necesaria esta información.

- **Etiqueta o *tag* del objeto seleccionado**

Muestra el nombre o etiqueta con la que se denomina el objeto seleccionado. Cada objeto tendrá una etiqueta distinta y única que lo identificará distinguiéndolo del resto de objetos. Esta etiqueta es el nombre utilizado a la hora de programar sus subrutinas.

- **Posición del objeto seleccionado:**

Muestra la posición y tamaño del objeto seleccionado.

- **Tamaño de la ventana**

Delimita el tamaño de la ventana de nuestra GUI.

- **Alinear objetos (*Align objects*)**

Mediante esta herramienta se podrán alinear los distintos objetos situados en el área de diseño mediante distintos criterios, tales como la alineación horizontal, vertical... entre otras. Es tan sencillo como seleccionar los objetos que se quieren alinear y ejecutar la mencionada herramienta.

- **Editor de menú *(Menu editor)***

Se emplea para gestionar los menús que se quiere que aparezcan en la interfaz gráfica, pudiendo añadir, borrar, agrupar, renombrar y modificar cada uno de ellos.

- **Editor de orden de etiqueta *(Tab order editor)***

Ajusta la lengüeta y el orden de apilamiento de los componentes existentes en el diseño con el que se esté trabajando.

- **Editor de barras de herramientas**

Permite crear barras de herramientas en las que se pueden incluir iconos representativos de funciones como *New File, Open, Save, Zoom, Print*, etc.

- **Editor del fichero M *(M-file editor)***

Permite ejecutar secuencialmente un programa (línea a línea) que recoja múltiples instrucciones. Es indicado para depurar programas y detectar fallos de programación de un modo más sencillo.

- **Propiedades de los objetos *(Property inspector)***

Mediante este icono se accede al *Property Inspector*, con el que se pueden variar las propiedades de cada uno de los objetos, las cuales se detallan en la Tabla 1. También se puede acceder al mismo pulsando con el botón derecho del ratón sobre el elemento y seleccionando la opción *Property Inspector* como se muestra en la Figura 42.

Tabla 1.- Propiedades de los componentes gráficos (botones, menús,…) utilizados para el diseño de la interfaz

PROPIEDAD	DESCRIPCIÓN
BackgroundColor	Color de fondo del objeto
BusyAction	Rutina de interrupción
ButtonDownFcn	Rutina al presionar el objeto
Callback	Función del objeto
CData	Imagen mostrada en el objeto
Children	Los *uicontrol* no tienen *children*
CreateFcn	Rutina ejecutada durante la creación del objeto
DeleteFcn	Rutina ejecutada durante el borrado del objeto
Enable	Habilita o deshabilita el objeto
Extent	Posición rectangular (solo lectura)
FontAngle	Inclinación de la letra

FontName	Tipo de fuente
FontSize	Tamaño de letra
FontUnits	Unidades del tamaño
FontWeight	Grosor de letra
ForegroundColor	Color del texto
HandleVisibility	Permite ocultar el identificador (*handle)*
HitTest	Permite que se seleccione por ratón
HorizontalAlignment	Alinea el texto de los objetos
Interruptible	Rutina de interrupción
KeyPressFcn	Rutina al presionar tecla del teclado
ListboxTop	Índice de texto superior en la lista
Max	Valor máximo
Min	Valor mínimo
Parent	Puntero al contenido del objeto (*Parent*)
Position	Vector de posición y tamaño del objeto
Selected	Muestra si el objeto está seleccionado
SelectionHighlight	Resalta el objeto cuando se selecciona
SliderStep	Paso de la barra (*slider*)
String	Texto o etiqueta del objeto
Style	Tipo de control (botón o *slider*)
Tag	Etiqueta del objeto
TooltipString	Descripción emergente asociada al objeto
Type	Clase del objeto gráfico
UIContextMenu	Menú de contexto asociado al objeto
Units	Unidades del vector *Position*
UserData	Datos especificados
Value	Valor del objeto
Visible	Visibilidad del objeto

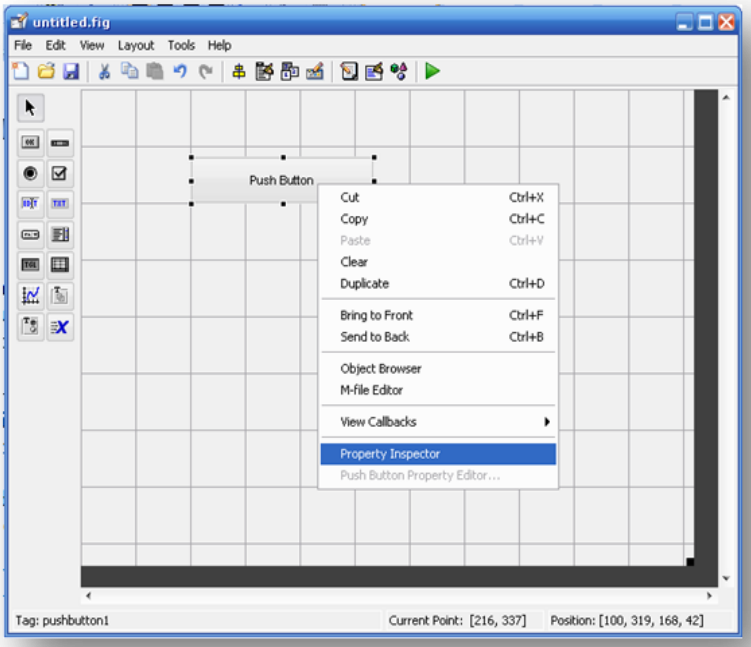


Figura 42.- Acceso al *Property Inspector*

Los componentes se pueden crear en la GUI como se describe anteriormente, o por medio de código usando el comando *uicontrol*. Este comando crea un componente específico configurado a partir de las propiedades o características que se introducen como argumentos del comando *uicontrol*.

Para ello se escribe el código *uicontrol('Propiedad',Propiedad,...)* por ejemplo *uicontrol('Style',Style,...)*. La propiedad 'Style' puede variar entre: *checkbox, edit, frame, listbox, popupmenu, pushbutton, radiobutton, slider, text y togglebutton*. Cada uno creará su correspondiente tipo de componente. Se pueden editar las demás propiedades del componente del mismo modo, *uicontrol('Propiedad1', Propiedad1 ,'Propiedad2',Propiedad2,…)*. Estas propiedades vienen explicadas en la Tabla 1.

Hay ciertos tipos de componente que se tienen que crear únicamente introduciendo un comando propio. Para crear tablas se utiliza el comando *uitable* y para paneles el comando *uipanel* y se editan las propiedades del mismo modo que con el comando *uicontrol*.

La propiedad *Parent* permite referenciar ciertos datos a un objeto. Esto permite, por ejemplo, pintar en un panel de la GUI los datos obtenidos en una función que es llamada dentro de la GUI, pero no está desarrollada dentro de la misma. Este proceso empieza obteniendo el *handle* o puntero del componente donde se pretende mostrar la información, por ejemplo un panel. A la función que proporciona los datos se le pasa como argumento el *handle* del panel. Dentro de la propia función se le indica que lo que pinte, lo haga en el *Parent*, que en el ejemplo será el *handle* del panel. De esta manera se direcciona la información y posibilita usar un mismo panel para mostrar distintas funciones.

- **Navegador de objetos *(Object browser)***

Ofrece un resumen de los objetos incluidos en la guide en forma de lista, de modo que se puedan localizar y gestionar de manera intuitiva y sencilla.

- **Grabar y ejecutar *(Run figure)***

Guarda y ejecuta la GUI con la que se esté trabajando.

3.2.1 Componentes de las GUI

Los componentes de la GUI son los controles para actuar sobre el programa, introducir información que necesita el programa o las gráficas o tablas donde el programa mostrará la información. Son parte fundamental para la comunicación entre el usuario final y el programa o herramienta.

En la parte izquierda de la ventana de diseño de la GUI (Figura 41) se encuentra la tabla de componentes. Este es el espacio desde el cual se seleccionan los componentes que se precisan de acuerdo con las diferentes funcionalidades que éstos presentan. Se puede acceder directamente o mediante el menú *Layout editor*.

En la Figura 43 se muestra un ejemplo de algunos de los objetos empleados.

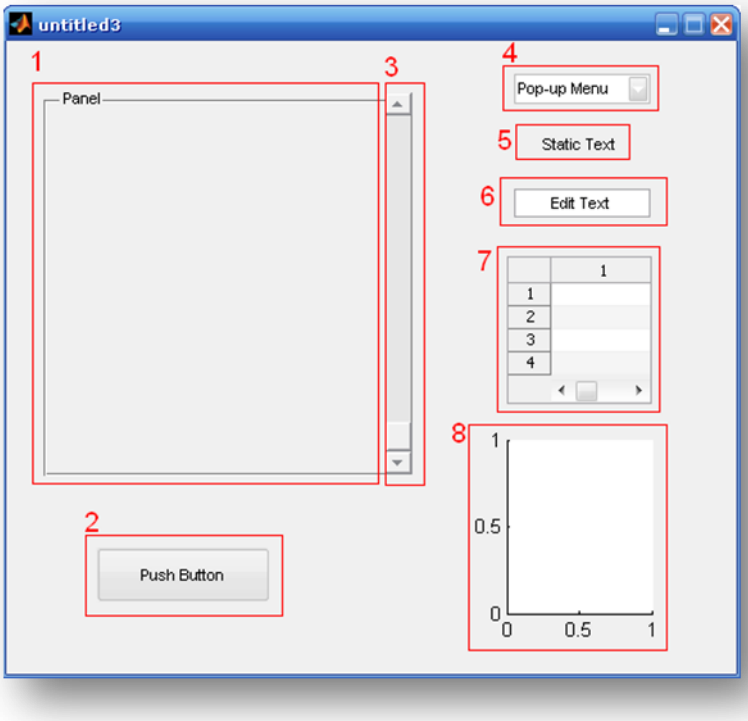


Figura 43.- Ejemplo de una GUI con múltiples objetos

En el ejemplo se emplean los siguientes objetos:

1. *Panel button*
2. *Push button*
3. *Slider*
4. *Pop-up menú*
5. *Static text*
6. *Editable text*
7. *Table*
8. *Axes*

Los componentes disponibles se detallan a continuación:

- ***Check box***

  Una casilla de verificación o *check box* es un elemento de interacción de la interfaz gráfica de usuario que permite hacer selecciones múltiples de un conjunto de opciones. Consta de una casilla que permite dos estados distintos, marcado y desmarcado. La marca implica la aceptación de la afirmación que va enlazada a ella, y por consiguiente, la ausencia de marca implica la negación de la afirmación. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *checkbox.*

- ***Editable text***

  Se trata de un cuadro de entradas en el cual se pueden introducir datos en formato texto. Posteriormente, el programa podrá obtener la información que se haya

introducido en los cuadros. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *edit.*

- **Menú *Pop-up***

El menú contextual (también llamado acceso directo contextual, y el menú emergente o *pop-up*) es un menú en una interfaz gráfica de usuario (GUI) que se despliega a partir de la interacción del usuario. Un menú contextual ofrece un conjunto limitado de opciones que están disponibles en el estado actual, o de contexto, del sistema operativo o de la aplicación. Por lo general, las opciones disponibles son las acciones relacionadas con el objeto seleccionado. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *popupmenu.*

- ***List box***

Un *listbox* es un control GUI que permite al usuario seleccionar uno o más elementos de una lista contenida dentro de un cuadro estático con múltiples líneas de texto. El usuario hace clic dentro del objeto para seleccionar una opción y a veces en combinación con la tecla de desplazamiento (*Shift*) o la tecla de control (*Ctrl*) con el fin de hacer selecciones múltiples. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *listbox.*

- ***Push button***

Se trata de un botón que, al ser pulsado, invoca el evento al que está asociado inmediatamente. Los botones suelen representarse como rectángulos con una leyenda o con un icono dentro, generalmente con efecto de relieve. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *pushbutton.*

- ***Radio button***

Un botón de opción o botón de radio de acción (a veces llamado incorrectamente "botón radial") es un tipo de control GUI que permite al usuario elegir una opción de un conjunto predefinido. Su utilidad es equivalente a la de *check box*. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *radio.*

- ***Toggle button***

Se trata de un botón con dos estados, que son encendido y apagado (*on* y *off*). Su funcionamiento es equivalente al de un interruptor. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *togglebutton.*

- ***Slider***

Se trata de una barra de desplazamiento para representar un rango de valores mediante el desplazamiento de los mismos por medio del ratón. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *slider.*

- ***Static text***

Se emplea para poner etiquetas mediante una cadena de caracteres, mostrando una cadena de texto (*string*) en el interior de un recuadro. Para elegirlo en la propiedad de estilo (*Style*) debe indicarse el valor *text.*

- ***Panel button***

Se emplea para agrupar una serie de botones relacionados entre sí por medio de un contexto común en un sólo grupo. También es posible agrupar conjuntos de gráficas en este tipo de componentes.

- ***Button group***

Permite exclusividad de selección con los *radio button* mediante el agrupamiento de los mismos.

- ***Table***

Permite crear tablas en las que se pueden cargar y representar datos de tipo numérico o alfanumérico

- ***Axes***

Permite unos ejes de coordenadas en los que la se pueden mostrar datos de modo gráfico o cargar imágenes.

3.3 Programación de la interfaz

Una vez finalizado el diseño de la interfaz gráfica de usuario en cuanto a componentes y propiedades de los mismos, al salvar los resultados se generará un archivo de texto automáticamente, cuya extensión es *.m.*

El archivo *.m* que se crea tiene una estructura predeterminada. Consta de un encabezado de inicialización y a continuación viene el código correspondiente a las siguientes subrutinas. Esto se entenderá mejor poniendo un ejemplo, como puede ser una aplicación cuyo archivo *.fig* consiste en 3 botones.

Se generará un archivo *.m* con una estructura inicial como la siguiente:

- Código de inicialización del programa, que no se edita:

```
function varargout = Prueba(varargin)
% End initialization code - DO NOT EDIT
```

- Código que se ejecuta al hacerse la ventana visible:

```
function Prueba_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
```

- Código cuya salida se muestra en la línea de comandos:

```
% --- Outputs from this function are returned to the command line.
function varargout = Prueba_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
```

varargout{1} = handles.output;

- Subrutina o *callback* del botón 1:

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

- Subrutina o *callback* del botón 2:

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

- Subrutina o *callback* del botón 3:

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)

Este código ha sido generado de forma automática por MATLAB al salvar el diseño gráfico de la guide.

Conviene detenerse en la siguiente instrucción, perteneciente al encabezamiento de la subrutina de, por ejemplo el botón 1.

function pushbutton1_Callback(hObject, eventdata, handles)

Los dos ficheros (*.fig* y *.m*) se relacionan a través de las subrutinas conocidas como *callback*.

Un *callback* es una función que se escribe y se asocia con un componente específico de la GUI, o con la misma GUI. El *callback* controla el comportamiento o la funcionalidad del componente o de la GUI completa, realizando alguna acción en respuesta a un evento. El evento puede ser un clic del ratón en un botón, menú, tecla, etc. Normalmente estas rutinas aparecen en el archivo M-file después del código de inicialización y la creación de los componentes.

Cuando ocurre un evento en un componente, MATLAB invoca al *callback* del componente asociado a ese evento. Por ejemplo, una GUI que contiene un botón que activa la visualización de datos en una gráfica. Cuando el usuario presione el botón, el software llama al *callback* asociado a pulsar ese botón, y entonces el *callback*, previamente programado, coge los datos y los muestra.

Existen distintos tipos de *callback*, asociados a las distintas acciones posibles de cada componente. Cada componente tiene *callback* específicos, ya que cada uno tiene un funcionamiento distinto. Por ejemplo un botón (*push button*) tiene cinco tipos de *callback*:

- *ButtonDownFcn:* Se ejecuta cuando el usuario pulsa el botón del ratón cuando el puntero está a menos de cinco pixeles del componente.

- *Callback:* Se ejecuta cuando el usuario pulsa el botón o menú.

- *CreateFcn:* Función que configura el componente cuando es creado. Se ejecuta después de que se cree el componente, pero antes de que se haya mostrado.

- *DeleteFcn:* Realiza funciones de 'limpieza' justo antes de que el componente se elimine.

- *KeyPressFcn:* Se ejecuta cuando el usuario presiona una tecla del teclado y el *Callback* del componente está seleccionado.

Existe una gran variedad de *callback* específicas para cada componente. Por defecto, en el *M-file* no suelen aparecer todos los *callback* pero se puede hacer que aparezcan seleccionando el componente en el GUIDE y activando los *callback* en el menú de Propiedades (*Property Inspector*).

Para una mejor comprensión de la comunicación entre ambos archivos se muestra en la Figura 44 una GUI llamada *untitled* con la correspondencia entre los elementos gráficos y sus *callbacks* asociados. Se pueden observar tres partes del código diferenciadas: el código de inicialización, del que se habla al principio de este capítulo y los *callbacks* de cada botón.

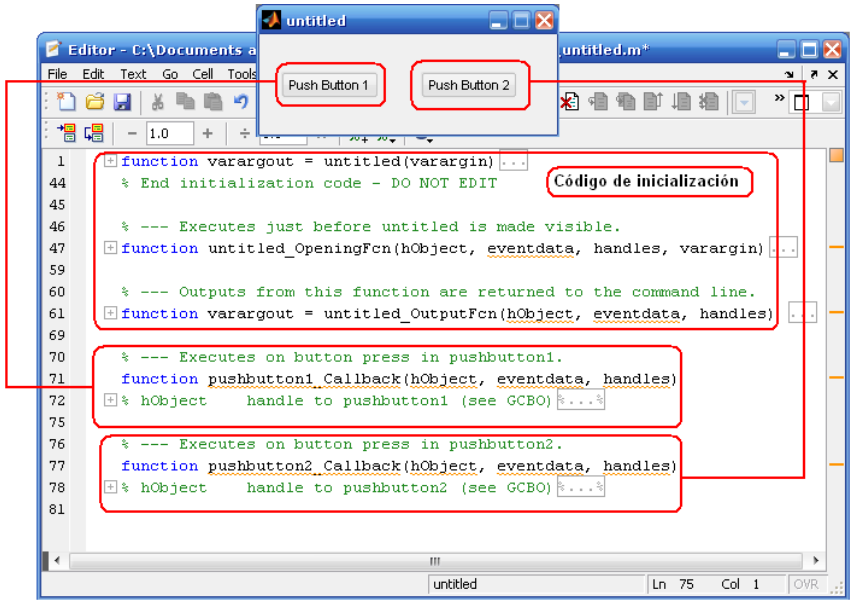


Figura 44.- Esquema de comunicación entre la parte gráfica y la parte de programación en una GUI con dos botones

Una vez que se graba la GUI, desde la consola de comandos de MATLAB se puede ejecutar el programa resultante únicamente escribiendo el nombre del archivo con extensión *.m*. Por ejemplo si se guarda un archivo de una GUI denominado como *ejemplo.fig* y *ejemplo.m*, respectivamente, escribiendo *ejemplo* en la línea de comandos y presionando *enter* se ejecuta el programa. Se puede ejecutar también pulsando el botón *Run* en la barra de herramientas en la ventana del Editor de MATLAB.

Por defecto, MATLAB otorga nombres a los elementos (en el ejemplo: *pushbutton1*), los cuales corresponden al valor de estilo del elemento en cuestión (*pushbutton*) con subíndice correspondiente al orden en el que se han ido agregando los elementos (el segundo botón está identificado como *pushbutton2* y así sucesivamente).

3.3.1 Manejo de datos entre los elementos de la aplicación y el archivo.m

Todos los valores de las propiedades de los elementos incluidos en la GUI (descritas en la Tabla 1) y los valores de las variables transitorias del programa se guardan en una estructura de tipo MAT, capaz de almacenar matrices, cadenas de caracteres (*strings*), registros estructurados, vectores, etc., a los que se accederá mediante un único puntero.

El nombre del puntero se asigna en el encabezado del archivo *.m*; tomando por ejemplo el programa listado anteriormente el puntero se asigna en:

handles = guihandles(fig);

handles es el puntero a los datos generado por MATLAB de la aplicación que se está diseñando.

Esta definición de puntero se salva con la siguiente instrucción:

guidata(fig, handles);

Por lo que *guidata*, es entonces la función para salvar los datos de la aplicación en el puntero de datos *handles*.

Instrucciones get y set

La asignación de valores de los componentes y la obtención de dichos valores se realiza mediante las sentencias *get* y *set*.

- **Instrucción *get***

Se trata de una instrucción con la cual se podrán leer los valores introducidos por el usuario.

get (h) devuelve todas las propiedades del objeto gráfico identificado por el *handle* h y sus valores actuales. Para esta sintaxis, h debe ser un escalar.

get (h, 'PropertyName') devuelve el valor de la propiedad *PropertyName* del objeto gráfico identificado por h.

Quizás sea preciso llevar a cabo alguna transformación de tipo de variable antes de memorizarlo para que los datos se puedan tratar correctamente a posteriori. Para entenderlo mejor, se recurre al siguiente ejemplo:

texto=str2double(get(handles.text1,'String'));

Mediante *get* se lee el valor de la casilla en la que se escribe el número de muestras que desea el usuario; a continuación, mediante *str2double*, se transforman los datos de tipo string a tipo double y se memorizan en la variable muestras.

- **Instrucción *set***

Es el comando mediante el cual se puede dar un valor determinado a un campo al desencadenarse un determinado evento, que puede ir desde la pulsación de un botón al proceso correspondiente a la ejecución de una determinada subrutina.

set (H, "PropertyName", PropertyValue,...) establece las propiedades (*PropertyName*) con los valores (*PropertyValue*) especificados del objeto identificado por el *handle* H. H puede ser un vector de *handles*, en cuyo caso se establecen todos los valores de las propiedades para todos los objetos.

Puede darse el caso expuesto anteriormente de que sea necesario cambiar el formato de los datos, por lo que se procederá de la misma forma que en el ejemplo anterior:

set(handles.text2,'String',num2str(100));

En éste caso se transforman los datos en formato numérico a una cadena de caracteres, y se da dicho valor *(100)* al campo *text2*.

- **Leer un fichero**

A continuación se muestra el código necesario para cargar un fichero (.mat) con los contenidos de señales anteriormente adquiridas u otros resultados del procesamiento que se han guardado, cualesquiera que sean los tipos de datos. Para ello se emplea la función *uigetfile.*

uigetfile abre un cuadro de diálogo que muestra los archivos en el directorio actual y permite que el usuario seleccione o escriba el nombre del archivo que se abrirá. Si el nombre del archivo es válido y si el archivo existe, *uigetfile* devuelve el nombre de archivo y su ruta cuando el usuario hace clic en Abrir. De lo contrario *uigetfile* muestra un mensaje de error y se devuelve el control al cuadro de diálogo anterior. El usuario puede introducir otro nombre de archivo o hacer clic en Cancelar. Si el usuario hace clic en Cancelar o cierra la ventana de diálogo, *uigetfile* devuelve 0.

A continuación se propone un ejemplo de programa que representa el comportamiento y el correcto uso de la función *uigetfile.*

```
[filename, pathname] = uigetfile( ...
{'*.mat', 'Archivos .mat (*.mat)'; ...
'*.*','Archivos .mat'}, ...
'Load data');

% If "Cancel" is selected then return
if isequal([filename,pathname],[0,0])
return

% Otherwise construct the fullfilename and Check and load the file.
else
end
```

- **Abrir una nueva ventana**

Esta opción es muy importante en MATLAB GUI ya que, por norma general, al llevar a cabo el desarrollo de una interfaz de usuario gráfica, se suele llamar a otras aplicaciones desde la ventana correspondiente a la aplicación principal. Se consigue de éste modo ir subdividiendo el trabajo a realizar entre distintas aplicaciones dependientes de la principal, de modo que ésta llame a unas u otras en función de los requerimientos del usuario de la interfaz.

Cada ventana tendrá asociado un fichero de código que se ejecuta mediante llamadas desde el menú principal. Esto es debido a que, si se tratara de integrar todas las funcionalidades de una aplicación en tal sólo un documento .m, se elevaría exponencialmente la cantidad de errores en la programación.

Todo lo descrito anteriormente se lleva a cabo con tan sólo una línea de código. Es sencillo y resulta una buena opción a la hora de programar interfaces gráficas con MATLAB GUI.

Como ejemplo se muestra una llamada desde la ventana principal a dos archivos de extensiones *Patrones.m* y *Patrones.fig* respectivamente.

Patrones();

Al pulsar el botón o seleccionar la opción que active este apartado del código del programa principal, se llamará a Patrones, abriéndose dicha aplicación en una nueva ventana. Se pueden editar algunos campos previos a la creación de la ventana; por ejemplo, la siguiente llamada creará la ventana patrones con el nombre 'Nombre de la ventana':

Patrones('Name', 'Nombre de la ventana');

3.3.2 Comando de ventanas de mensaje

Existe una serie de comandos por defecto que crean ventanas de tipo informativo. Son de gran utilidad ya que permiten transmitir información al usuario de forma rápida cuando intente realizar una operación no permitida o sea necesario informarle de que cierta acción ha comenzado o se está finalizando. Los mensajes son de la siguiente forma:

- **Mensaje de aviso (Figura 45)** : *warndlg('Esto es un aviso', ' Aviso');*

Figura 45.- Mensaje de aviso

- **Mensaje de error (Figura 46):** *errordlg('Esto es un mensaje de error',' Error ');*

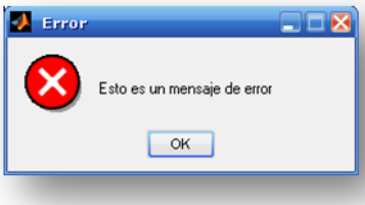


Figura 46.- Mensaje de error

- **Mensaje de ayuda (Figura 47):** *helpdlg('Esto es una ayuda',' Ayuda ');*

Figura 47.- Mensaje de ayuda

- **Mensaje informativo (Figura 48)**: *msgbox('Esto es un cuadro de mensaje', ' Info ');*

Figura 48.- Mensaje informativo

- **Pregunta (Figura 49):** *questdlg('Esto es una pregunta',' Pregunta ');*

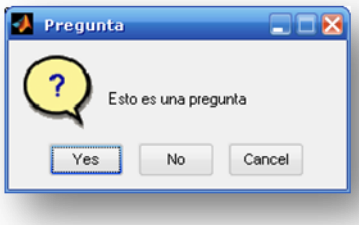


Figura 49.- Mensaje de pregunta

En este último caso se lee la respuesta que seleccione el usuario, de manera que se limitan las posibles acciones guiando al usuario por el programa de una forma correcta.

3.4 Datos tipo *cell*

El tipo de datos empleado en la base de datos utilizada por las funciones será en su mayoría de tipo *cell*.

Un *Cell Array* es una colección de contenedores llamados *cells* o celdas en los que se puede almacenar datos de diferentes tipos. Su utilización como base de datos es de gran utilidad ya que permite mezclar distintos tipos de datos en un mismo contenedor. La Figura 50 muestra una matriz *cell* de 2 por 3. Las celdas de la primera fila contiene una matriz de enteros sin signo, una matriz de líneas de texto y una matriz de número complejos. En la segunda fila hay tres tipos de matriz, siendo la última una *cell* anidada:

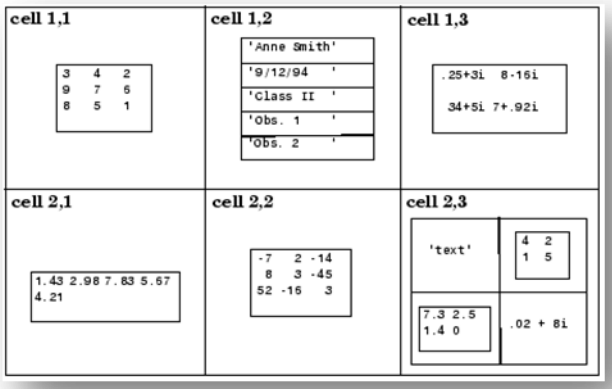


Figura 50.- Ejemplo de archivo tipo *cell*

Cada celda de la matriz de *cells* contiene algún tipo de dato de MATLAB. Cada dato en estas matrices pertenece a una de las clases definidas por el usuario o a tipos de datos MATLAB, y pueden tener cualquier dimensión válida; esto incluye desde matrices de 1 por 1 (matriz escalar), a matrices con una o más dimensiones iguales a cero (matriz vacía).

La capacidad de agrupar matrices de distintos tipos de datos y tamaños es la característica más significativa de los datos tipo *cell*.

Existen una serie de comandos que permiten crear archivos tipo cell, transformar los datos que contiene u obtener información de estos archivos. Estos se describen en la tabla de funciones relativa a archivos de tipo cell (Tabla 2).

Tabla 2.- Funciones relativas al manejo y creación de archivos de tipo *cell*

Propiedad	Definición
cell	Crea matriz de tipo *cell*.
cell2mat	Convierte una array de tipo *cell* en un array de tipo numérico.
cell2struct	Convierte una array de tipo *cell* en un array de tipo *struct*.
celldisp	Muestra el contenido de la *cell*.
cellfun	Aplicar funciones a cada celda del array de tipo *cell*.
cellplot	Muestra gráficamente la *cell*.
cellstr	Crea un array de tipo *cell* a partir de un array de tipo string.
class	Determinar la clase del objeto.
deal	Distribuir las entradas a las salidas.
isa	Determinar si la entrada es un objeto de la clase dada.
iscell	Determinar si la entrada es una matriz de tipo *cell*.
iscellstr	Determinar si la entrada es una matriz de tipo *cell* cuyo contenido es de tipo string.
isequal	Comprueba si las matrices son idénticas.
mat2cell	Convertir matriz a matriz de tipo *cell* con celdas de tamaño variable.
num2cell	Convertir matriz numérica a matriz de tipo *cell* con las celdas de un tamaño constante.
struct2cell	Convertir archivo tipo *struct* a matriz de tipo *cell*.

Para operar con archivos tipo *cell* en los que el contenido es de tipo numérico o alfanumérico (como es el caso de los datos suministrados en la fase de desarrollo de la aplicación, sección 2.3.1) se aconseja transformar los datos de la matriz de tipo *cell* (celdas) a tipo matriz. Se han empleado principalmente las funciones *cell2mat* (Figura 51), *mat2cell* y *num2cell*.

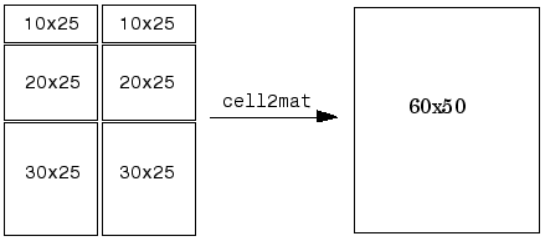


Figura 51.- Gráfica del funcionamiento de la función *cell2mat*

Se puede acceder a cualquier celda únicamente indicando la fila y la columna correspondientes (*Ej: A{1,1}, accede al contenedor situado en la primera fila primera columna*). Si se trata de un contendor que contiene otro tipo de contendor, el acceso a los datos de este último se realiza con el siguiente comando: A{i,j}(m,n), siendo A la *cell* principal, *i y j* los índices fila(i) y columna(j) correspondientes a la celda en la que se encuentra el contenedor y *m* y *n* los índices fila(m) y columna(n) en los que se encuentra el dato que se quiere consultar.

3.5 Herramienta *selectdata*

Para poder seleccionar datos en una gráfica en MATLAB se necesita una herramienta de selección gráfica. La herramienta *selectdata* [8] permite al usuario seleccionar datos de una gráfica utilizando el ratón. Permite realizar la selección de distintos modos y obtener una lista de los índices de los puntos seleccionados. La función tiene los siguientes argumentos:

[pointslist,xselect,yselect] = selectdata(varargin)

3.5.1 Argumentos de entrada

Los argumentos de entrada (*varargin*) a la función *selectdata* serán siempre pares de nombre de propiedad y valor de la misma.

La función consta de los siguientes argumentos o propiedades de entrada con los que se configuran los modos de selección: *'Action', 'Axes', 'BrushSize', 'Identify', 'Ignore', 'Pointer'* y *'SelectionMode'*

- ***'Action'** - {'list', 'delete'}*

Acción que realizará la herramienta a la hora de devolver los datos. Si la propiedad elegida es '*delete'* elimina los puntos seleccionados en la gráfica después de la selección final. En caso de ser '*list*' devuelve una lista de los índices de los puntos, es decir la posición del punto contada de izqda. a dcha.

Valor por defecto: *'list*'

- ***'Axes'*** - puntero de la gráfica que se quiere procesar.

Denota la gráfica en la que se seleccionaran los puntos.

Valor por defecto: *gca,* correspondiente a la gráfica actual o la última que se modificó.

- ***'BrushShape'** - {'rect', 'circle'}*

Fija la forma del puntero para la selección ya sea rectangular (*'rec*t') o circular ('*circle'*). Ambas opciones van asociadas a los ejes, esto implica que si la gráfica no es exactamente cuadrada, la selección con circulo será elíptica.

Esta opción solo es válida cuando la propiedad *SelectionMode* es de tipo *'brush'*.

Valor por defecto: *'circle'*

- ***'BrushSize'*** - Intervalo válido: Tamaño del puntero entre 0 y 0,25

Determina el grosor del puntero de tipo *'brush'*

Opción solo válida cuando la propiedad *SelectionMode* es de tipo *'brush'*.

El valor por defecto especifica un puntero rectangular que tiene un 5% de las dimensiones de los ejes de la gráfica.

Valor por defecto: 0,05

- ***'Identify'*** - *{'on', 'off'}*

Los datos seleccionados serán realzados temporalmente en color rojo a medida que se seleccionen.

Valor por defecto: *'on'*

- ***'Ignore'*** – Ignora un grupo de datos

Lista de punteros de datos a ignorar. Esto permite usar la herramienta *selectadata* solo en algunos grupos de puntos, mientras otros de la misma gráfica son ignorados. Esta es una opción muy útil en los casos en los que se representan varios conjuntos de datos y funciones en una sola gráfica, donde todos ellos aparecen mezclados. Utilizando la opción de ignorar podremos seleccionar sobre cuál de los conjuntos queremos realizar la selección gráfica.

Los puntos a ignorar se introducirán del siguiente modo*: [punterodato1 punterodato2… punterodatoN]*.

Valor por defecto: [], conjunto vacío o ningún dato ignorado.

- ***'Label'*** - *{'off', 'on'}*

Crea etiquetas de texto con los valores de x e y que aparecerán en cada punto seleccionado.
Si se selecciona un gran número de puntos, crear las etiquetas puede consumir mucho tiempo y recursos. Esta opción es de utilidad para selecciones de pocos puntos.

Valor por defecto: *'off'*

- ***'Pointer'*** - *{'crosshair' | 'fullcrosshair' | 'arrow' | 'ibeam' |'watch' | 'topl' | 'topr' | 'botl' | 'botr' | 'left' | 'top' | 'right' | 'bottom' | 'circle' | 'cross' | 'fleur' | 'hand' }*

Cambia el cursor del puntero cuando se activa la selección. Al terminar la selección el puntero recupera su configuración anterior.

Valor por defecto: *'crosshair'*

- ***'Return'*** - *{'selected' | 'unselected'}*

Permite que los datos que devuelva la función puedan ser los puntos seleccionados o bien los puntos no seleccionados.

Valor por defecto: *'selected'*

- ***'SelectionMode'*** *- {'Lasso', 'Brush', 'Rect', 'Closest'}*

Si se utiliza *'Brush'*, entonces la selección se realizará mediante un círculo cuyo tamaño se define por medio de la propiedad *'BrushSize'*. El centro de este círculo será el puntero del ratón. Para seleccionar los datos, mantener el botón izquierdo del ratón pulsado y arrastrar el puntero por la gráfica. Cada punto que atravesemos se seleccionará y se resaltará en rojo

Para utilizar el modo de selección *'Lasso'* se pulsa y arrastra con el ratón en la gráfica creando una región que encierra los puntos de interés. La opción '*Lasso*' permite crear áreas con perímetros curvilíneos, no necesariamente circunferencias, así como polígonos. Los puntos dentro de esta región serán los puntos seleccionados. Al realizar varios lazos superpuestos, o que se intersequen entre sí, la herramienta no selecciona de forma correcta los puntos incluidos dentro de estos lazos.

Si se utiliza *'Rect'*, pulsando el botón izquierdo del ratón y, arrastrándolo, se crea una región rectangular dentro de la cual se encontrarán los puntos seleccionados

Si se utiliza *'Closest'*, con un único clic del ratón en la gráfica se obtiene el punto más cercano al puntero del ratón. Se puede mover el puntero pulsando el botón izquierdo del ratón y desplazándolo por la gráfica y se irán resaltando los puntos seleccionados.

Valor por defecto: *'Lasso'*

- ***'Verify'*** *- { 'off' | 'on' }*

Si la propiedad elegida es *on*, se creará una ventana de dialogo después de la selección. Da la opción de rehacer la selección, aceptar la selección o simplemente cancelar la selección, en cuyo caso no será seleccionado ningún punto.

Valor por defecto: *'off'*

3.5.2 Argumentos de salida

La herramienta proporciona distintos tipos posibles de salidas en función de la gráfica en la que estemos trabajando.

- *Pointslist* - Lista de los índices de los puntos seleccionados. Si solo hay un grupo de puntos representados, estos aparecerán en un único vector. Si hay múltiples grupos de puntos en la gráfica, los índices se entregarán en un array de tipo *cell*.

- *xselect* - matriz (o matriz *cell* array) que contiene las coordenadas X de los puntos identificados en la selección.

- *yselect* - array (o *cell* array) que contiene las coordenadas X de los puntos identificados en la selección.

xselect e *yselect* únicamente se obtendrán si se indica a la función que devuelva esos datos. Por defecto aparecerá la lista con los índices de los puntos seleccionados.

3.5.3 Ejemplos de uso de la herramienta

A continuación se incluyen dos ejemplos para mostrar la utilidad práctica de esta herramienta en gráficas con un solo grupo de datos y con múltiples grupos.

3.5.3.1 Ejemplo 1: Grupo simple de puntos. Tipo *`brush´*

Representación de un grupo de puntos. Selección de alguno de ellos con el ratón usando un puntero rectangular. Obtención de los índices de los puntos seleccionados.

- **Código:**

Crear los datos a representar.

```
x = 0:.1:1;
y = x.^2;
```

Representar en una gráfica de puntos (Figura 52) x en abscisas e y en ordenadas.

```
plot(x,y,'o')
```

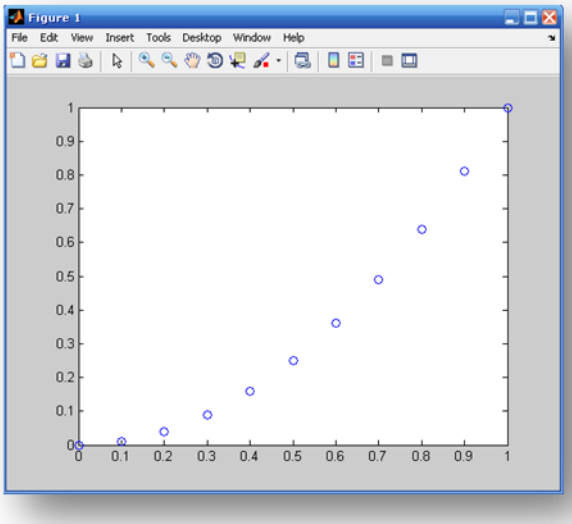


Figura 52.- Representación gráfica de los puntos para el ejemplo 1 de utilización de la herramienta *selectdata*

Activar la herramienta de selección configurando el tipo de selección en '*brush*'. Almacenar tras la selección de los puntos (Figura 53) los datos en la variable *pl*.

pl = selectdata('selectionmode','brush');

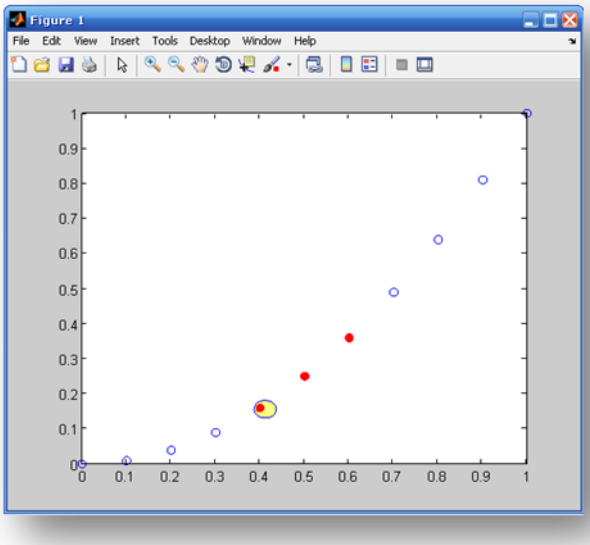


Figura 53.- Selección de puntos en el ejemplo 1 de utilización de la herramienta *selectdata*

Los datos obtenidos se almacenan en la variable *pl* que contiene los índices de los puntos seleccionados (Figura 54):

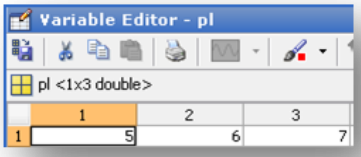


Figura 54.- Datos almacenados en la variable *pl* tras la selección en el ejemplo 1

3.5.3.2 Ejemplo 2: Múltiples grupos de puntos. Tipo *`Lasso´*

Representación de una curva y datos individuales en una sola gráfica. Selección de los datos individuales ignorando la curva, aun cuando esta esté seleccionada

- **Código:**

Crear los datos a representar:

```
x = 0:.01:1;
y = exp(x);
ynoisy = y + randn(size(y))/2;
```

Representar en una única gráfica la curva con el nombre de puntero *h1* y los datos individuales con el nombre de puntero *h2* (Figura 55)*:*

```
h1 = plot(x,y,'-');
hold on
h2 = plot(x,ynoisy,'o');
```

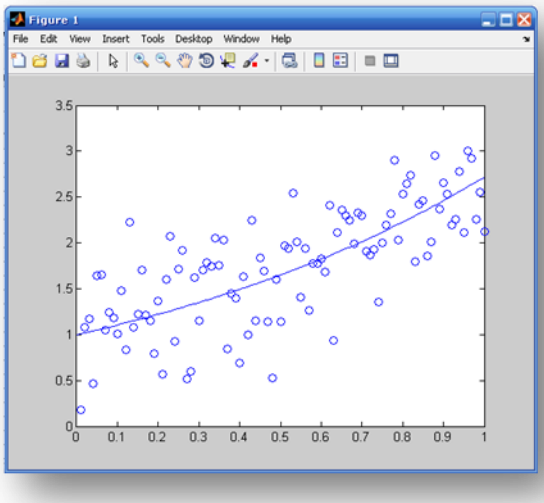


Figura 55.- Representación gráfica de los puntos para el ejemplo 2 de utilización de la herramienta selectdata

Obtener la matriz de tipo *cell* con la información de los datos seleccionados (Figura 56), con los índices de los datos en la variable *pl* y sus coordenadas x e y (Figura 57). En la

configuración de la selección elegir el tipo *lasso* e ignorar los datos que corresponden a la variable h1:

[pl,xs,ys] = selectdata('sel','lasso','ignore',h1);

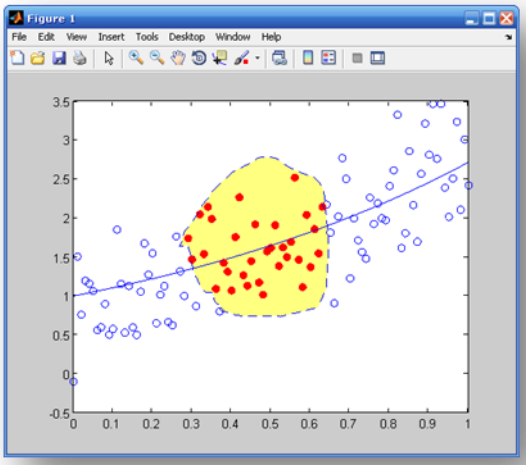
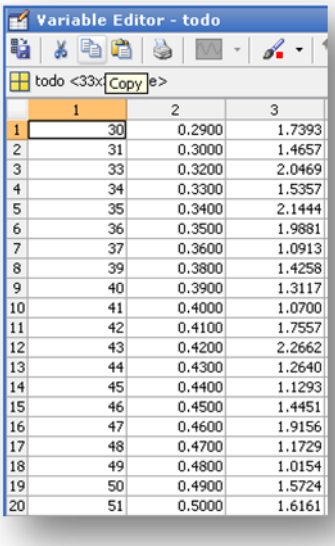


Figura 56.- Selección de puntos en el ejemplo 2 de utilización de la herramienta selectdata

En la Figura 57 se muestra una variable *todo* que contiene en la primera columna los índices de los puntos seleccionados, en la segunda columna las coordenadas x de cada uno de los puntos seleccionados y en la tercera columna las coordenadas y de los puntos seleccionados



	1	2	3
1	30	0.2900	1.7393
2	31	0.3000	1.4657
3	33	0.3200	2.0469
4	34	0.3300	1.5357
5	35	0.3400	2.1444
6	36	0.3500	1.9881
7	37	0.3600	1.0913
8	39	0.3800	1.4258
9	40	0.3900	1.3117
10	41	0.4000	1.0700
11	42	0.4100	1.7557
12	43	0.4200	2.2662
13	44	0.4300	1.2640
14	45	0.4400	1.1293
15	46	0.4500	1.4451
16	47	0.4600	1.9156
17	48	0.4700	1.1729
18	49	0.4800	1.0154
19	50	0.4900	1.5724
20	51	0.5000	1.6161

Figura 57.- Datos almacenados en la variable *todo* tras la selección en el ejemplo 2

Capítulo 4

Estructura de la aplicación PDtool

4.1 Introducción

Este capítulo describe la solución adoptada al problema propuesto. Para ello se explic*a*rá el proceso de creación de la GUI, el tipo de programación que se ha elegido y un esquema del funcionamiento de la herramienta así como las funciones que se han tenido que crear para poder cumplir la tarea.

4.2 Metodología de desarrollo

Antes de empezar a programar es imprescindible hablar con el usuario final de la GUI para poder comprender cuáles son las necesidades exactas que tiene que cubrir la aplicación a diseñar.

Para conseguir este fin es necesario entender el tipo de datos y variables que son introducidas por el usuario, así como las funciones de procesamiento desarrolladas por un experto. Esta información se ha descrito previamente en los capítulos 2 y 3. Es preciso también conocer el modo en el que el usuario quiere que se presenten los datos resultantes cuando la aplicación se está ejecutando.

Para un correcto ritmo de trabajo se ha optado por seguir el siguiente sistema de diseño, una simplificación del recogido por Mayhew en [9] 'extraído del curso de interfaces de usuario del OCW de la UC3M [10] en el que se distinguen tres bloques básicos (Figura 58): análisis de requisitos, diseño y depuración de errores e instalación final.

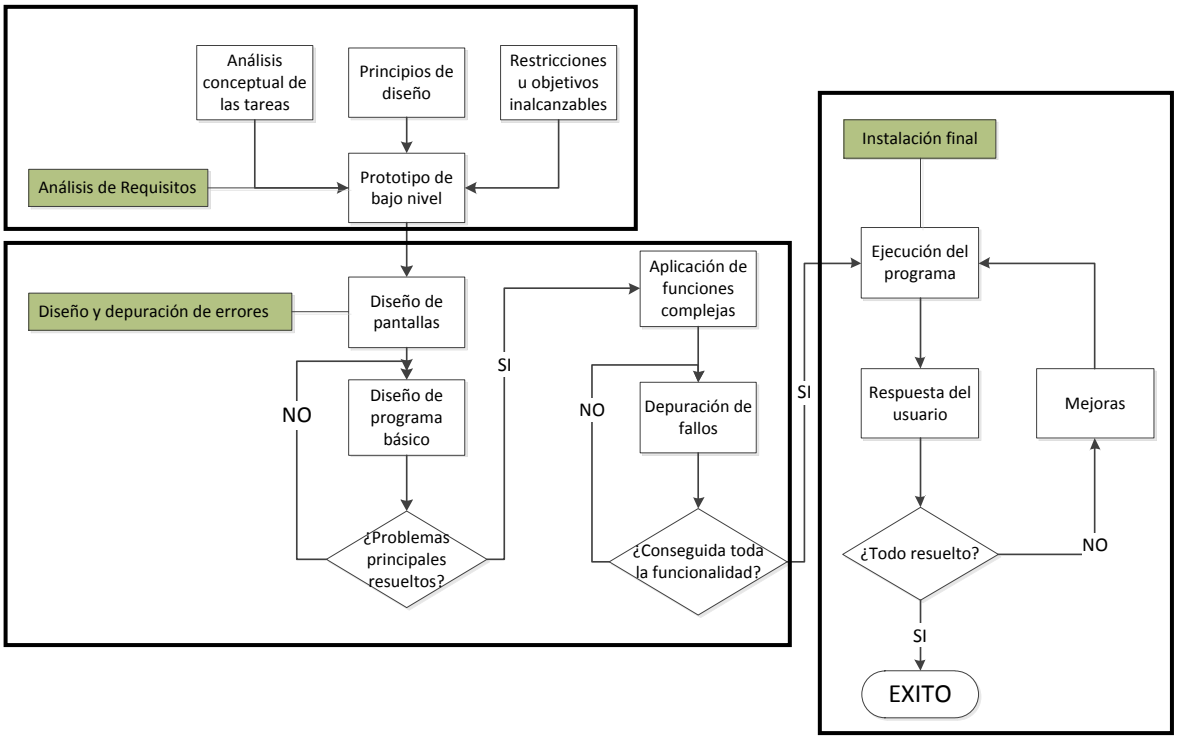


Figura 58.- Esquema de trabajo para el diseño de una interfaz

Para diseñar correctamente una GUI se suele comenzar con un primer borrador, por lo general realizado a mano alzada sobre un papel, llamado Prototipo de Bajo Nivel [ANEXO 12.2], en el que representar un boceto con el cual explicar al usuario las ideas que se tienen para llevar a cabo en el proyecto, escuchando las suyas propias, dando lugar a una realimentación de conceptos y puntos de vista que puede llevar al cambio de las ideas preconcebidas en un principio por el diseñador. Este es el bloque superior izquierdo de la Figura 58 con el nombre de Análisis de requisitos.

De esta forma se consigue minimizar los imprevistos en el desarrollo de la herramienta, ya sean por imposibilidades derivadas de objetivos inalcanzables o por otro tipo de restricciones para el proyecto. Se consigue además que el usuario se sienta parte activa del proyecto, ya que en él estarán depositadas parte de sus ideas y preferencias, aumentando de este modo la satisfacción final con el resultado obtenido.

Una vez que se tienen claras las especificaciones, restricciones y objetivos perseguidos a satisfacer por la interfaz de usuario, es necesario hacer un programa muy básico con igual funcionalidad que la GUI que se quiere diseñar. Esta parte del diseño corresponde con el bloque inferior izquierdo de la Figura 58 . Se puede segmentar cada una de las acciones en programas más pequeños y simples para abordar los problemas de una forma más rápida. La filosofía por la que se ha optado es desarrollar el software básico para cumplir las tareas fundamentales, y poco a poco ir añadiendo complejidad a la funcionalidad del programa sin perder en ningún momento el fin de dicha aplicación.

Antes de incorporar el programa a la aplicación es necesario hacer todo tipo de pruebas con él, planteando casos fuera de lo usual, poniendo a prueba al programa buscando el máximo de estabilidad posible y depurando los fallos encontrados hasta que se esté completamente seguro de que el programa que se va a incorporar a la GUI es el adecuado.

Una vez que se tenga guardado el programa completo se podrá incorporar a la herramienta, dando funcionalidad a las pantallas y a sus controles, de modo que se pueda comprobar su correcto funcionamiento.

Tras comprobar que se han resuelto los principales problemas y se ha alcanzado la total funcionalidad del programa, el usuario dará su opinión final. En caso de que el usuario final proponga alguna nueva ampliación o mejora se trabajará en la última versión del programa, siendo la mejora de tipo menor ya que previamente se han satisfecho las necesidades especificadas.

4.3 Requisitos básicos de diseño de la interfaz

Se utilizó la bibliografía [2] y entrevistas [ANEXO 12.1] para sentar las bases necesarias para crear la herramienta que cumpla con todos los requisitos. Los requisitos de diseño tratan aspectos técnicos de la interfaz y también alguna característica general. Los principales requisitos técnicos propuestas por el usuario final son:

- Correcta gestión de las bases de datos y de la memoria, necesaria ya que MATLAB no borra por defecto cierto tipo de datos y esto consume recursos de forma innecesaria.

- Todas las ventanas deberán ser configurables, y todas tendrán la opción de cerrar la ventana o de reiniciar todos los valores.

- Los datos se deben mostrar en la GUI intentando minimizar el número actual de ventanas que se crean al ejecutar las funciones.

- Guardar los datos para posibles análisis posteriores.

Otros requisitos que ha de cumplir la interfaz son las especificaciones de usabilidad. Estas especificaciones tratan aspectos tales como:

- **Efectividad:** Se refiere a como de bien un sistema hace lo que se supone que debe hacer.

- **Eficiencia:** Se refiere a la forma en que un sistema ayuda a los usuarios a llevar a cabo sus tareas.

- **Seguridad:** Se refiere a que el usuario está protegido de condiciones peligrosas y de situaciones indeseables: incidencia de los errores al realizar las actividades. Si el usuario comete un error, ¿puede recuperar la información?

- **Utilidad:** Se refiere a que el sistema proporciona el tipo de funcionalidades correctas, de manera que el usuario puede hacer lo que necesita y lo que quiere hacer.

- Que se pueda **aprender fácilmente**: Se refiere al esfuerzo que requiere aprender a usar un sistema (regla de los 10 minutos [10]).

- Que sea **fácil de recordar** cómo se usa: Se refiere al esfuerzo que requiere recordar un sistema después de que se haya aprendido como se usa y no se haya utilizado durante un tiempo. Incluye proporcionar ayudas para su posterior uso.

4.4 Requisitos de funcionalidad de la interfaz

En las entrevistas con el usuario final [ANEXO 12.1] se determinaron los requisitos que debe cumplir el programa completo y se realizaron los prototipos de bajo nivel. Se definieron siete submenús específicos para cada parte del análisis.

4.4.1 Requisitos de la interfaz

Estos requisitos se centran en la funcionalidad de cada una de las herramientas o submenús que contiene la aplicación.

- **Ventana Principal:** Esta ventana tiene que contener botones que proporcionen acceso a las siete aplicaciones o submenús de la herramienta. Tiene que estar siempre visible junto con la aplicación que se haya seleccionado. Únicamente habrá una aplicación abierta a la vez, así se evita que existan procesos que se estén ejecutando innecesariamente.

- **Ventana Herramienta patrones:** En este submenú se visualizará la señal del patrón así como los datos numéricos de dicho patrón. Como configuración existe la opción de seleccionar el patrón con el que se buscará la descarga parcial en las señales adquiridas. También existe la posibilidad de añadir patrones nuevos creados a partir de una nueva base de datos o seleccionando la señal directamente en la gráfica. Estos nuevos patrones se podrán guardar.

- **Ventana Visualización de señales:** Se representarán las señales adquiridas durante el experimento de localización acústica de descargas parciales en una

única ventana. La configuración permitirá cargar la señal que se seleccione y sus canales de forma individual.

- **Ventana Activación del motor de búsqueda:** Se utilizarán las funciones diseñadas para la detección de descargas parciales. En este submenú se configurará el tipo de detección que se va a utilizar, se fijará el intervalo de la señal donde se realiza la búsqueda y se podrán determinar ciertos aspectos de configuración más avanzada. Dispondrá de un botón de activación que al pulsarlo active la búsqueda configurada. Se mostrará el progreso de la búsqueda por medio de una barra de progreso. Al finalizar la búsqueda almacenará los datos obtenidos en una base de datos.

- **Ventana Visualización de las señales encontradas:** Se representará gráficamente el resultado de nuestra búsqueda y se podrá seleccionar con un botón de zoom la zona deseada. En función de cada tipo de detección se mostrará una información distinta.

- **Ventana Visualización de la base de datos:** Se mostrarán las bases de datos generadas tras las detecciones.

- **Ventana Herramientas estadísticas:** Se compararán los datos de la detección en una gráfica y en la configuración se podrá elegir los valores a contrastar, delimitar que parte de la señal se usará o la relación que hay entre los canales. Habrá un botón para representar ese análisis y otro para seleccionar datos de interés.

- **Ventana Localización:** Se cargarán los datos obtenidos de las búsquedas y en la configuración se elegirá la búsqueda y localización de las descargas usando distintos métodos.

- Se han definido también unos **objetivos adicionales** cuyo cumplimiento no es preceptivo, pero sí deseable. Estos son distintos usos de una herramienta de selección de datos dentro de la gráfica. En el caso de la Herramienta patrones, la herramienta de selección debe permitir escoger una ventana y almacenarla. Para el caso de la Herramienta estadística es importante poder seleccionar un grupo de puntos y el valor de dichos puntos. Estos puntos tienen que ir asociados a una descarga parcial.

Se han propuesto algunas ideas propias aceptadas por el usuario final. Las principales ideas propuestas son:

- **Versión bilingüe:** Se propuso una versión en español y en inglés. Las traducciones se han realizado consultando al usuario experto la traducción al español de ciertos términos técnicos.

- **Aspectos de representación:** Hay ciertos aspectos de la representación que se han cambiado para adaptarlos a una sola ventana. Por ejemplo, inicialmente los canales de las señales adquiridas se representaban en ventanas independientes. Se ha propuesto un conjunto de funciones que permiten representar las señales en una sola ventana y se ha adaptado para un máximo de 8 canales. Para una correcta visualización se ha añadido una barra que controla las señales que se muestran.

- **Localización por planos:** Se ha propuesto una modificación de las funciones de localización para poder desplazarse en la dirección Z de la cuba. Inicialmente las funciones mostraban numerosas ventanas emergentes con todos los cortes de la cuba que se analizaban.

4.4.2 Prototipos de bajo nivel

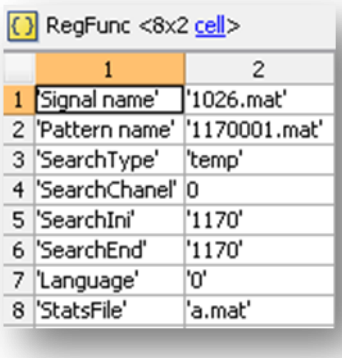
Tras el análisis de los requisitos y especificaciones se procede al diseño a mano alzada de los prototipos de bajo nivel. En ellos se describe a grandes rasgos el diseño gráfico que se considera óptimo distribuyendo el espacio para que pueda contener cada ventana los objetos necesarios para tener total funcionalidad. Estos prototipos de bajo nivel se recogen en el anexo correspondiente [ANEXO 12.2] situado al final de esta memoria.

4.5 Metodología de programación

Durante el diseño de la aplicación se ha optado por seguir un mismo proceso de comprobación y verificación de los datos que se van a mostrar para todas las ventanas. De esta manera se puede evitar que el programa realice alguna operación errónea y se podrá orientar al usuario en los pasos a seguir para un correcto funcionamiento, siendo éste el estándar a seguir cada vez que se realice cualquier tipo de acción.

4.5.1 Registro de variables globales

Se ha optado por una comprobación rutinaria de un archivo de tipo *cell* que contiene las variables globales para que el programa no pierda información y para su correcto funcionamiento. El archivo es *RegFunc.mat* (Figura 59) que se creará después de elegir el idioma en la pantalla inicial o portada.


RegFunc <8x2 cell>

	1	2
1	'Signal name'	'1026.mat'
2	'Pattern name'	'1170001.mat'
3	'SearchType'	'temp'
4	'SearchChanel'	0
5	'SearchIni'	'1170'
6	'SearchEnd'	'1170'
7	'Language'	'0'
8	'StatsFile'	'a.mat'

Figura 59.- Archivo de variables globales *RegFunc.mat*

Este archivo contiene la siguiente información:

- ***Signal name:*** Nombre (variable de tipo *string*) de la señal que se ha cargado en la herramienta Representar señales. Almacenar esta información permite al programa guardar el nombre de la señal representada anteriormente y si se cierra la ventana y se vuelve a abrir la misma no se perderá la información.

- ***Pattern name:*** Nombre (variable de tipo *string*) del patrón que se ha seleccionado en Herramienta patrones. Almacenar esta información permite al programa guardar el nombre del patrón seleccionado anteriormente y si se cierra la ventana y se vuelve a abrir la misma no se perderá la información. En procesos posteriores es necesario conocer esta información para realizar la búsqueda de DP de forma correcta.

- ***SearchType:*** Nombre (variable de tipo *string*) del tipo de búsqueda que se ha seleccionado en la herramienta Motor de Búsqueda. Se consideran válidos 4 posibles palabras:

  - *None*: Ningúna búsqueda realizada.
  - *1canal:* Búsqueda en un único canal.
  - *Global:* Búsqueda acústica conjunta.
  - *temp*: Búsqueda con referencia temporal.

  Se ha optado por el uso de palabras ya que son significativas en cuanto al proceso que se les atribuye. Para una versión posterior es más rápido asociar a su significado una palabra en el código que una variable de tipo numérico.

- ***SearchChanel:*** Canal (variable de tipo numérico) en el cual se realizará la búsqueda, cuando sea necesaria su especificación.

- ***SearchIni/SearchEnd:*** Señal inicial y final (variables de tipo numérico) del intervalo en el que se ha realizado la última búsqueda.

- ***Language:*** Variable de tipo numérico que permite traducir los textos de la herramienta al idioma seleccionado. Para idioma inglés, *Language* toma el valor 1. El 0 representa el idioma español.

- ***Statsfile*:** Nombre (variable de tipo *string*) del archivo en el que se ha guardado la selección de puntos utilizando la herramienta de selección en el submenú Herramienta estadística.

La Figura 60 muestra los pasos de comprobación genéricos que se dan cuando se abre cualquiera de los submenús.

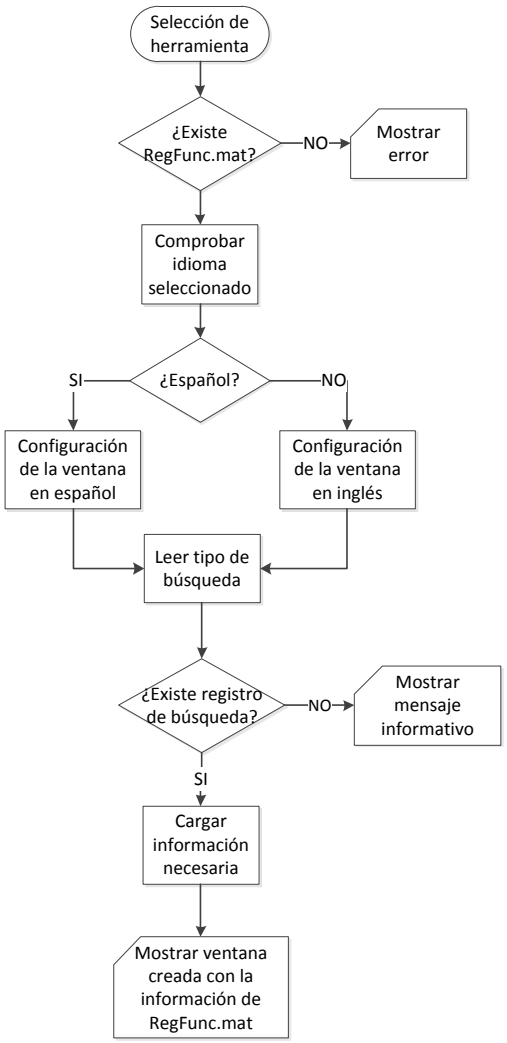


Figura 60.- Comprobación del idioma y el tipo de búsqueda por variables globales

Durante la creación de la ventana de cada submenú o herramienta el programa buscará el archivo *RegFunc.mat*. En caso de no encontrarlo el programa avisará con un mensaje de error, ya que se perdería la secuencia de información que necesita la herramienta para funcionar correctamente. Tras la comprobación de la existencia del archivo *RegFunc.mat*, lo común ha sido comprobar el tipo de búsqueda que se ha realizado. Esto es útil a la hora de crear las ventanas ya que son distintas para cada tipo de búsqueda, como se mostrará más adelante en el apartado *Desarrollo de la aplicación PDtool(4.6)* en este capítulo.

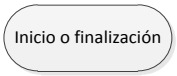
Una vez se haya comprobado el tipo de búsqueda se crea la ventana del submenú o la herramienta que se vaya a utilizar en función de ese tipo de búsqueda. Hay ocasiones que al intentar acceder a algún submenú, tras haber realizado una búsqueda en concreto, este

submenú no está activo para los datos generados por la búsqueda. Estos casos se describen en el apartado *Desarrollo de la aplicación PDtool* en este capítulo.

A la hora de cerrar la herramienta, el programa muestra un mensaje preguntando si se desea guardar la información. En caso afirmativo este archivo permanecerá en la carpeta raíz de la herramienta. Cuando se ejecute nuevamente la herramienta comprobará si existe o no un archivo *RegFunc.mat* en la carpeta raíz y de ser así informará de que hay datos guardados de una búsqueda anterior. En el caso negativo, borrará el archivo RegFunc.mat, creando uno nuevo la próxima vez que se ejecute la herramienta.

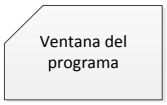
Ha sido de gran ayuda a la hora de programar ya que ha permitido simular estados del programa sin generarlos en realidad. Por ejemplo, puede simular que se ha realizado una búsqueda de tipo Global únicamente escribiendo en el archivo *RegFunc.mat* en la celda correspondiente a *SearchType* la palabra Global. De esta manera se puede actuar sobre el programa como si se hubiera realizado una búsqueda Global.

Los diagramas de flujo de cada uno de los diferentes programas se muestran en las páginas siguientes. Los símbolos utilizados para la representación de dichos diagramas son los siguientes:

- Inicio o finalización del programa: 

- Procesos del programa: 

- Entrada y salida de datos o archivos: 

- Ventanas que abre el propio programa: 

- Condiciones que se deben cumplir: 

4.6 Desarrollo de la aplicación PDtool

La aplicación final se ha llamado *PDtool* (herramienta DP en inglés). A continuación se muestra el esquema básico de la aplicación (Figura 61), basado en las especificaciones. A partir de este esquema se ha diseñado el sistema de ventanas de cada uno de los submenús.

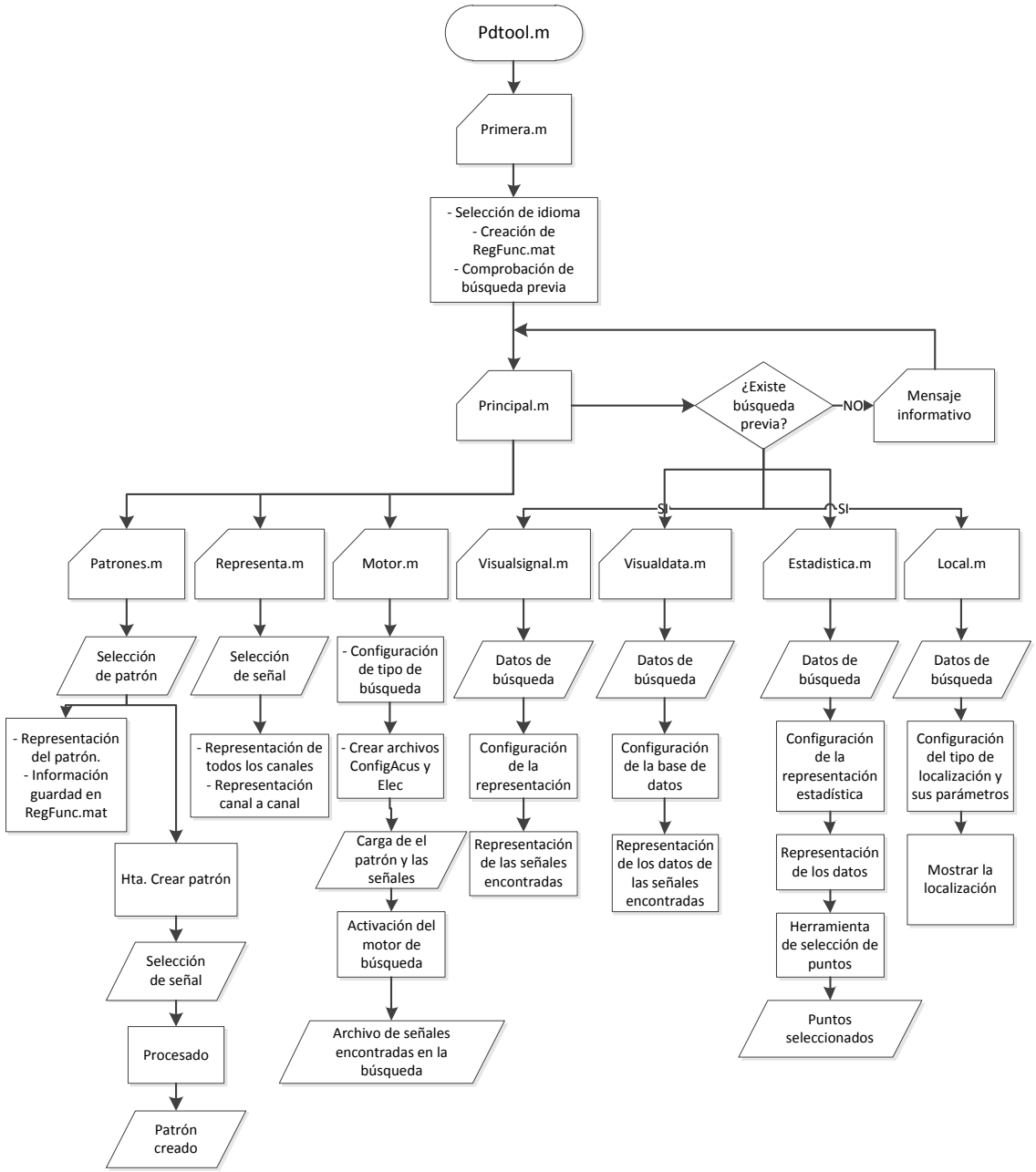


Figura 61.- Esquema general de la aplicación PDtool

Como se observa en la Figura 61, el esquema de la aplicación PDtool consta de siete submenús. Cada botón abrirá el submenú asociado. Para comprender el esquema de funcionamiento, se realiza un resumen del funcionamiento normal, basado en el análisis del experto en explicado anteriormente en el capítulo 2.

El primer paso es la selección de un patrón en la herramienta *patrones.m*. En caso de no existir patrón se puede crear a partir de una señal adquirida, que podremos encontrar más fácilmente con la herramienta *representa.m*. Tras seleccionar un patrón se accede a la herramienta *motor.m* donde podremos realizar la detección e identificación de señales DP. Es por eso que para acceder a estas herramientas no es necesario que se haya realizado ninguna búsqueda previa, ya que es el primer paso para todo el proceso.

Las herramientas *visualsignal.m, visualdata.m, estadística.m* y *local.m* dependen directamente de los datos generados tras una búsqueda, por lo que no se podrá acceder a ellas sin tener ningún archivo de registro generado por algún tipo de búsqueda.

La herramienta *local.m,* responsable de la localización espacial de las DP, no está siempre disponible ya que no es posible una localización si no se ha realizado una búsqueda con referencia temporal.

4.6.1 Proceso inicial

Para ejecutar la portada de la aplicación se ha creado una función llamado *PDtool.m* que será el comando que ejecute la herramienta. Esta función contiene los comandos responsables para añadir las carpetas necesarias para el correcto funcionamiento a nuestro espacio de trabajo de MATLAB (*Workspace*). El funcionamiento de *PDtool* se puede observar en la Figura 62.

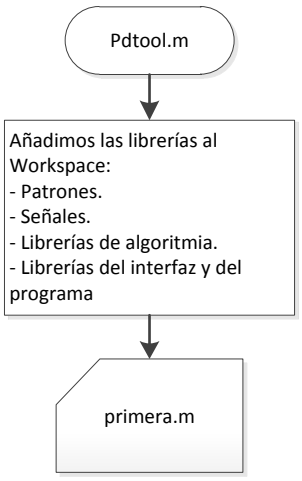


Figura 62.- Proceso de inicio de la aplicación y carga de las librerías de la aplicación PDtool

Se añaden las librerías creadas por Jesús Rubio Serrano que contienen el conjunto de funciones empleadas para la búsqueda y procesado de las DP con el comando *addpath 'Librerias'*, las librerías creadas por Pablo Grandas Aguado para la creación de la interfaz

(algunas de ellas son adaptaciones de las librerías de Jesús para la interfaz) con el comando *addpath 'ToolLibrerias'*, la carpeta que contiene los patrones y la que contiene las señales del mismo modo con *addpath 'Patrones'* para los patrones y *addpath 'Signals'* para la que contiene las señales.

Tras añadir las carpetas con las librerías y las señales se ejecuta el comando *primera.m* lo que ejecuta la portada de nuestro programa (Figura 63).

Figura 63.- Portada de la herramienta

Al seleccionar uno de los dos idiomas, se creará el archivo *RegFunc.mat* descrito anteriormente, en caso de que no se haya guardado en alguna búsqueda anterior. Se guardará el valor del idioma elegido en *RegFunc.mat* siendo 0 para el español y 1 para el inglés. Tras este proceso se cerrará la ventana de la portada y se abrirá la pantalla Principal (Figura 64)

Figura 64.- Menú Principal

Esta ventana consta de siete botones que corresponde a cada una de las herramientas o submenús de la herramienta *PDtool,* un botón de *Reset* y otro para Cerrar. Pulsando cada uno de los botones se abrirá el submenú correspondiente, por tanto se ejecutará el archivo .m asociado al submenú correspondiente. La relación entre los nombres de los botones y sus funciones .m asociadas es la que se describe en la siguiente tabla (Tabla 3), y se usará en el resto de la memoria para hacer referencia a cada uno de los submenús.

Tabla 3.- Archivos de programación de cada interfaz asociados a los distintos submenús

Nombre de submenú	Archivo *.m asociado
Herramienta patrones	*patrones.m*
Visualización de adquisiciones	*representa.m*
Activación motor búsqueda	*motor.m*
Visualización eventos encontrados	*visualsignal.m*
Visualización base de datos	*visualdata.m*
Herramienta estadísticas	*estadística.m*
Localización	*local.m*

Pulsando el botón *Reset* el programa borrará todos los archivos auxiliares creados por el programa así como las gráficas creadas. El botón cerrar permite elegir entre las opciones de borrar o guardar los datos generados por las búsquedas. En caso de elegir la opción de no guardar los datos, borrará los datos como lo hace el botón *Reset* y tras borrarlos cerrará la ventana. Ambos botones trabajan tal y como se ha definido en los requisitos y especificaciones del usuario final.

4.6.2 Desarrollo de la herramienta Patrones

Al pulsar el botón de Herramientas patrones se abre la ventana correspondiente (Figura 65) a la primera herramienta o submenú *patrones.m*, tras la comprobación rutinaria.

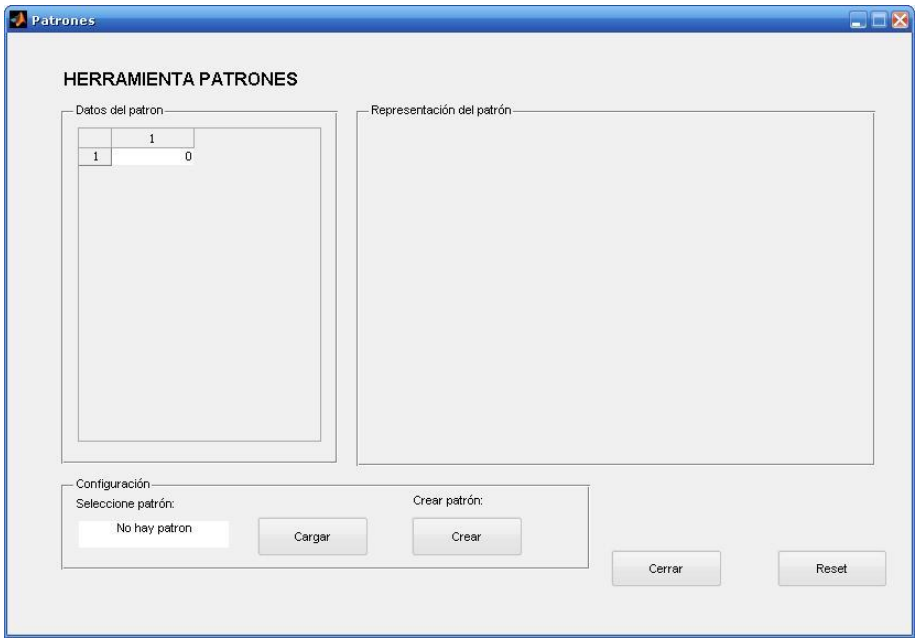


Figura 65.- Ventana diseñada para la herramienta Patrones

El funcionamiento de esta herramienta, representado en la Figura 69, se divide en dos procesos principales. El primero es el de cargar patrones existentes pulsando el botón Cargar. Para ello se tiene que disponer de las señales de tipo patrón (descritas en el capítulo 2). Una vez seleccionado un archivo de tipo patrón, el programa carga sus datos numéricos en la tabla creada. El archivo *RegFunc.mat* actualiza el nombre del patrón seleccionado. También se utilizan ciertas funciones para la representación gráfica de los patrones.

- **pintaunpatron(patron, hpanel):** Pinta un patrón sobre la señal en la que fue adquirida. Como argumentos, hay que indicar a la función el nombre del patrón en formato texto (Ej: "1170003.mat") así como el puntero al panel donde se va a pintar la gráfica. *pintaunpatron.m* es una modificación de la función *pintapatron.m* para que pueda pintar la gráfica generada por dicha función en el panel de la aplicación en lugar de en una ventana nueva.

- **imppatron(patron, hpanel):** Pinta únicamente la señal que corresponde al patrón del mismo modo que la función anterior.

El botón de *Reset* ejecuta un código que borra el panel, los datos de la tabla y restaura los valores iniciales de los componentes de la ventana. También se actualiza el nombre del patrón en el archivo *RegFunc.mat.*

El botón de Cerrar cierra la ventana borrando los datos gráficos.

El segundo proceso es el de crear patrón. Pulsando el botón Crear, se ejecuta la herramienta *HPatron.m.* Esta función crea una ventana (Figura 66) donde se configura la señal original de la cual se va a seleccionar el patrón (señal original, canal de dicha señal y tipo o naturaleza de la señal).

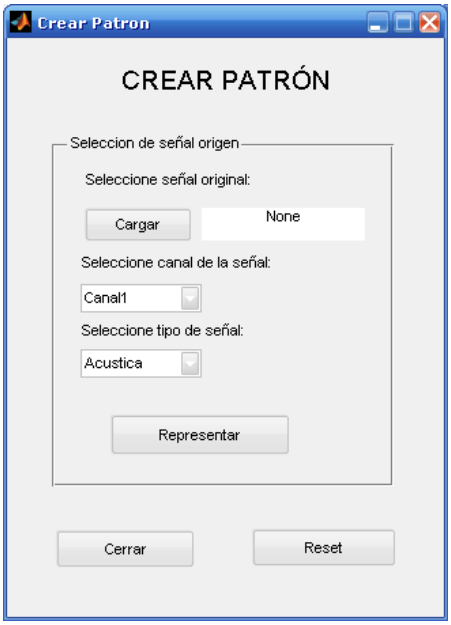


Figura 66.- Ventana diseñada para la herramienta Crear patrón

Al pulsar Representar en la ventana creada se ejecutará la función pintind2 y también se crea el archivo *prop.mat* que guarda las características configuradas en esta ventana. Este será un archivo que guie al proceso en la creación de un patrón. La función *pintaind2.m* crea la ventana (Figura 67) que contiene la señal elegida y dos botones (Seleccione ROI y Procesar).

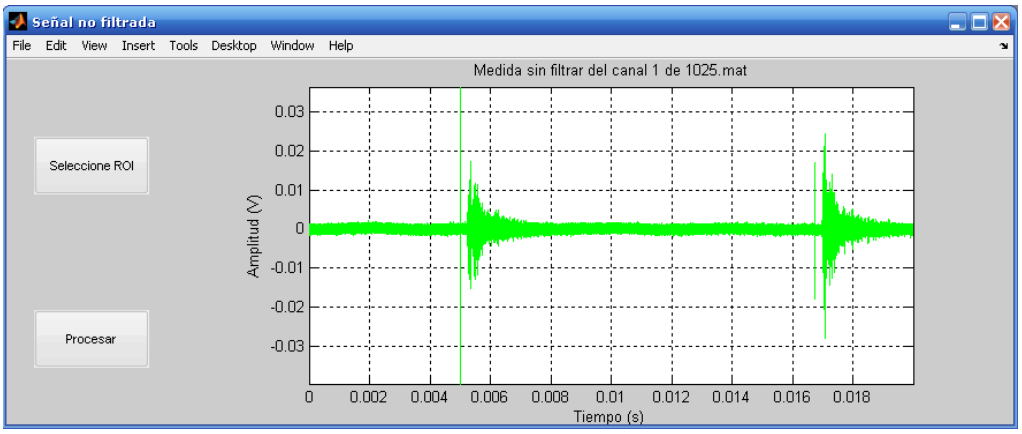


Figura 67.- Primer proceso de la creación de un patrón, donde se muestra la señal (no filtrada) elegida donde se seleccionará el patrón

- **Botón Seleccione ROI:** Ejecuta la rutina *primpatronizar.m* que activa la herramienta *selectdata.m* y permite seleccionar un grupo de puntos. Al realizar la selección la rutina guarda el índice del primer y el último dato de la selección en el archivo *selección.mat*. Tras realizar la selección podremos pinchar en el botón Procesar.

- **Botón Procesar:** Ejecuta la rutina *secpatronizar.m* que realiza un filtrado a la señal seleccionada mediante la función *patronizar.m*. Al terminar el filtrado y procesado de la señal seleccionada se pinta la señal filtrada en una nueva ventana (Figura 68) debajo de la ventana actual y contiene también un nuevo botón con la etiqueta Procesar patrón.

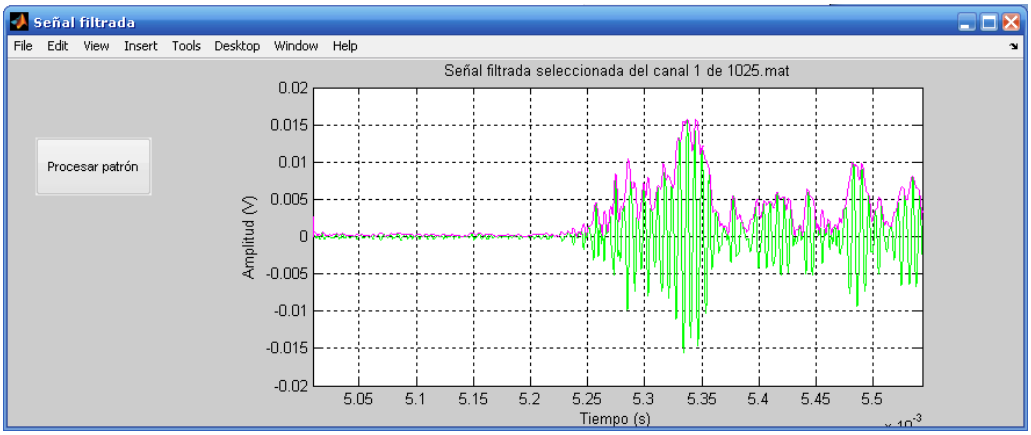


Figura 68.- Segundo proceso de la creación de un patrón, donde se ha filtrado la señal seleccionada previamente

Si se considera que la señal es adecuada para servir como patrón, se procesa la información pulsando el botón Procesar patrón. De no ser así se puede volver a seleccionar en la ventana superior

- **Botón Procesar patrón:** Ejecuta la rutina *tercpatronizar.m* que utiliza la señal seleccionada, filtrada, para crear el patrón final. Para ello se hace uso de la función *patronizarw.m*.

Tras realizar este proceso se muestra el patrón superpuesto a la señal original y también se crea el archivo de tipo patrón que contiene los datos del patrón.

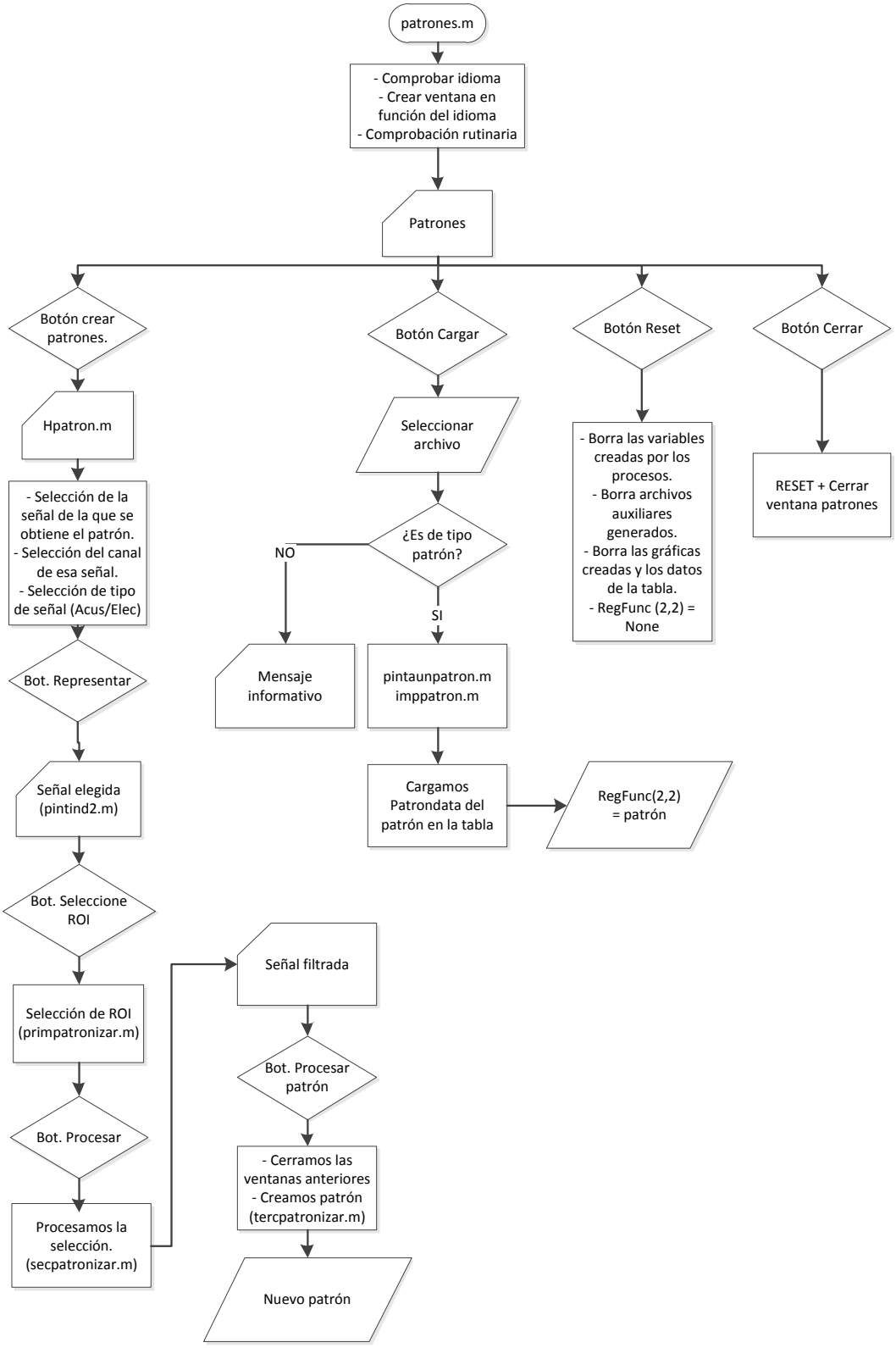


Figura 69.- Esquema de funcionamiento de la herramienta Patrones

4.6.3 Desarrollo de la herramienta Visualización de adquisiciones

Al pulsar el botón de Visualización señales se abre la ventana correspondiente (Figura 70) a la segunda herramienta o submenú *representa.m*, tras la comprobación rutinaria.

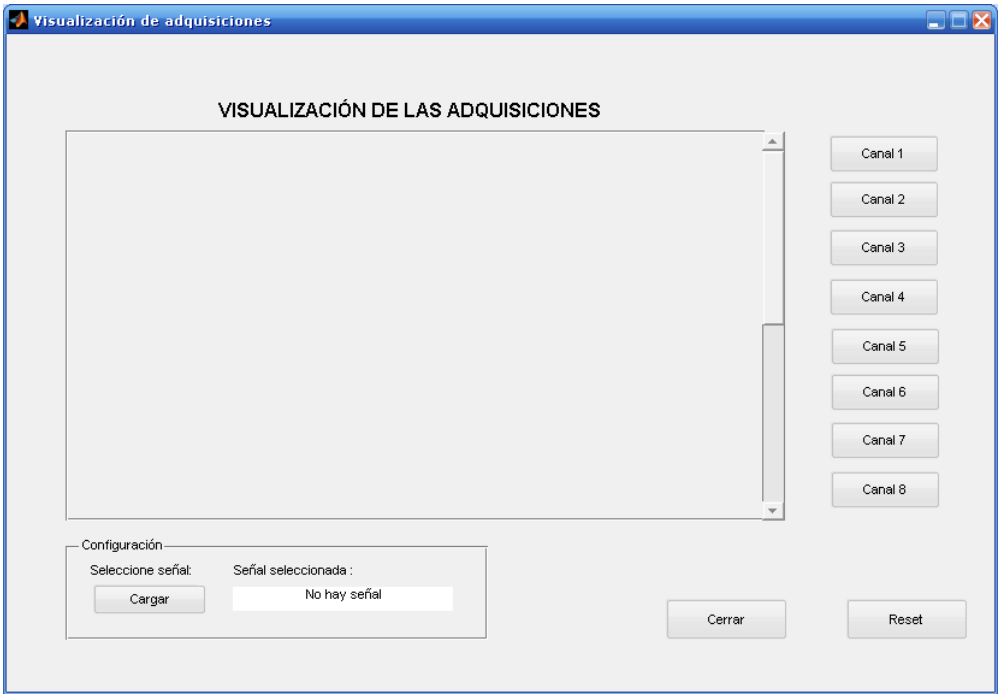


Figura 70.- Ventana diseñada para la herramienta de Visualización de adquisiciones

El funcionamiento de esta herramienta, representado en la Figura 71 se basa principalmente en un proceso. Dicho proceso es la carga de una señal. Pulsando el botón Cargar se seleccionará un archivo de tipo señal (descrito en el capítulo 2), se actualiza el archivo *RegFunc.mat* y se ejecutará la función *pintasenal.m*.

- **pintasenal(filename,hpanel,PosScroll):** Función que pinta una señal de ocho canales, dividiéndolos en dos pantallas de cuatro canales cada una. La función se basa en la posición de la barra de *scroll* para saber que señales hay que pintar. Los argumentos son el nombre de la señal (*filename*) en formato texto (Ej: "1170.mat"), el puntero al panel donde se va a representar la señal (*hpanel*) y la posición del *scroll* (*PosScroll*) que tendrá valor 1 cuando esté arriba y 0 cuando esté abajo. Esta es una ampliación de la función *pinta4ch.m* añadiendo la funcionalidad de la barra de *scroll*.

Una vez cargada una señal válida se pueden pulsar los botones situados en el lado derecho de la pantalla. Estos abren una pantalla con el canal de esa señal pintado individualmente ejecutando la función *pintind.m*.

- **pintind(filename,chanel):** Función que pinta un único canal de una señal en una ventana individual. Los argumentos son el nombre de la señal en formato texto (*filename*) y el canal que se desea pintar (*chanel*).

El botón de *Reset* borra el panel y restaura los valores iniciales de los componentes de la ventana. También se actualiza el nombre de la señal en el archivo *RegFunc.mat*.

El botón de Cerrar cierra la ventana borrando las gráficas creadas.

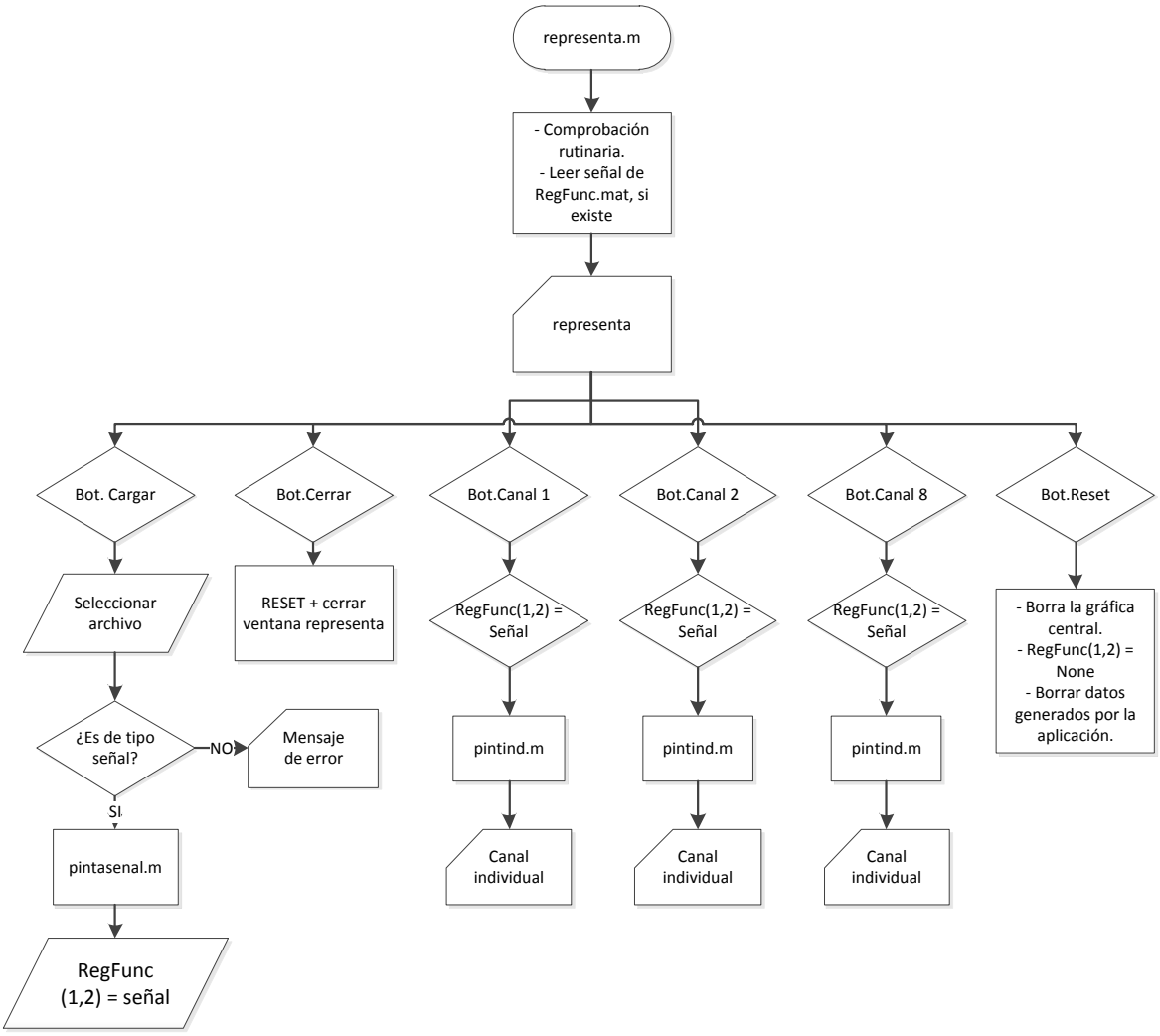


Figura 71.- Esquema de funcionamiento de la herramienta representa

4.6.4 Desarrollo de la herramienta Motor de búsqueda

Al pulsar el botón de Activación motor de búsqueda se abre la ventana correspondiente (Figura 72) a la tercera herramienta o submenú motor.m. Su funcionamiento se describe en la Figura 74.

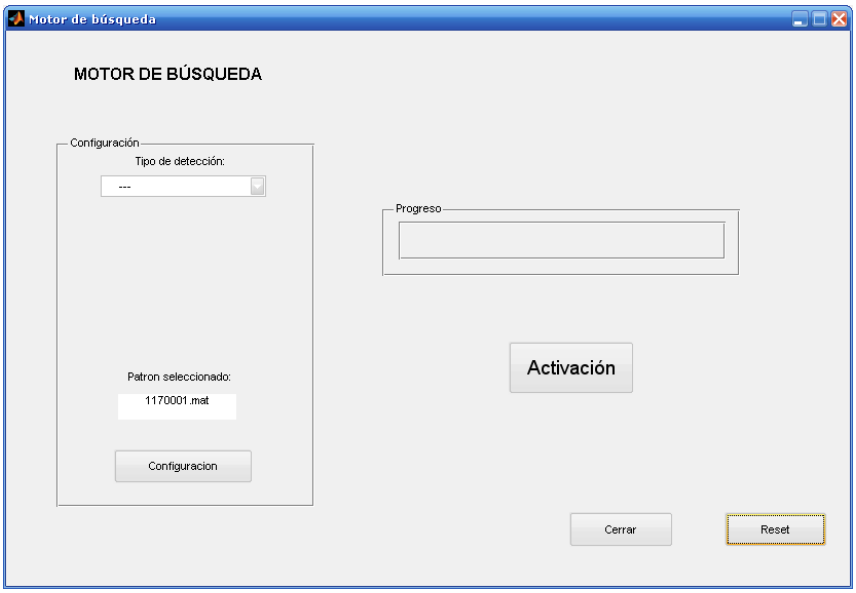


Figura 72.- Ventana diseñada para el motor de búsqueda

Esta ventana es auto configurable, lo que significa que, en función del tipo de detección que se seleccione, la ventana se adaptará para pedir únicamente los valores necesarios. Para ello hace aparecer o desaparecer ciertos campos para rellenar.

El proceso fundamental es el asociado al botón Activación. En función del tipo de detección que se seleccione, el botón Activación utiliza unas funciones u otras. Se han definido tres tipos de detección en el capítulo 2, por lo que el botón Activación tendrá tres posibles funciones, mostradas en el menú de Tipo de detección:

- **Búsqueda en un solo canal:** La función asociada a este tipo de búsqueda es *Signafinder1ch.*

- **Búsqueda global acústica:** Las funciones asociadas a este tipo de búsqueda son *Globalacusfinder.m + AcusPDmixingv2.m*

- **Búsqueda con referencia temporal:** Las funciones asociadas a este tipo de búsqueda son *Globalacusfinder.m seguida de Acuselecasoc1.m, 2.m y 3.m y por último DelayCalc.m.*

El botón Configuración permite crear los archivos de configuración básicos para poder realizar cualquier tipo de detección (*ConfigAcus.mat CofigElec.mat*). Esto es posible gracias a la herramienta *Configuracion_Motor.m* que despliega una ventana (Figura 73) en la que se puede configurar cualquier tipo de variable que influye en la búsqueda.

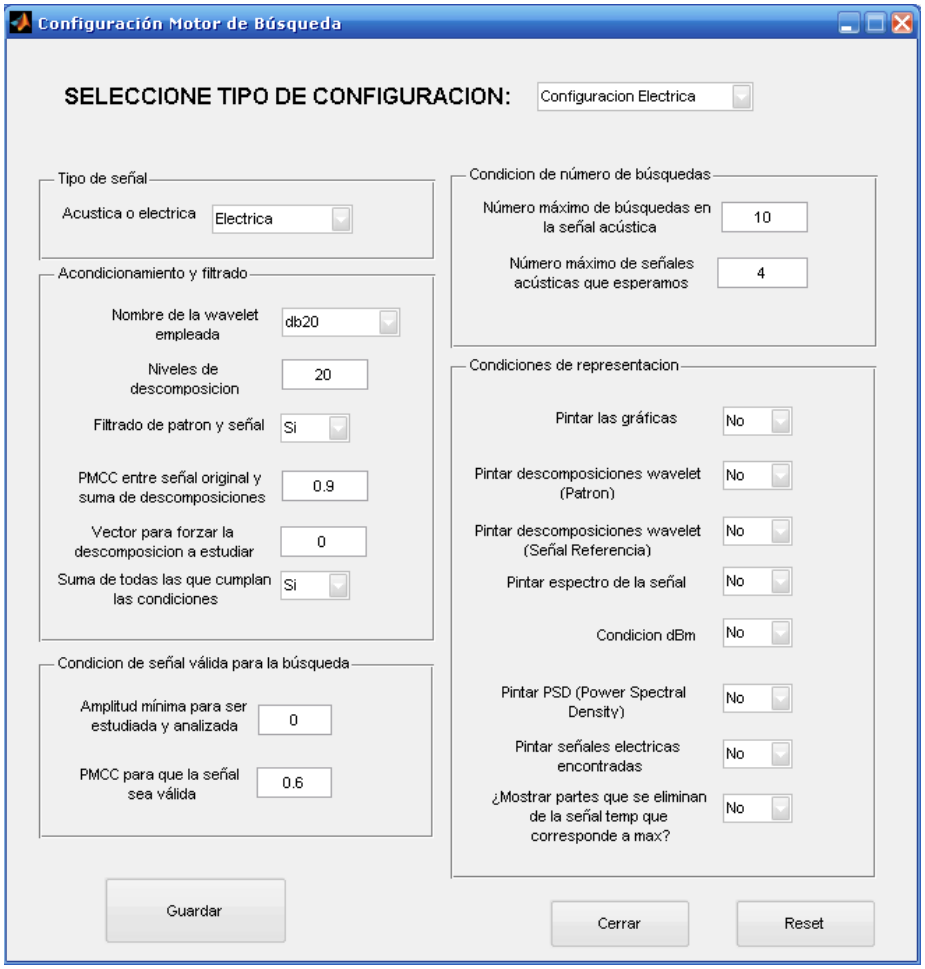


Figura 73.- Ventana diseñada para la herramienta *Configuracion_Motor.m* encargada de la configuración de las funciones

El botón de *Reset* borra restaura los valores iniciales de los objetos de la ventana. También se actualiza los valores del intervalo de señales analizadas en el archivo *RegFunc.mat*. El botón de Cerrar cierra la ventana y borra las variables del programa.

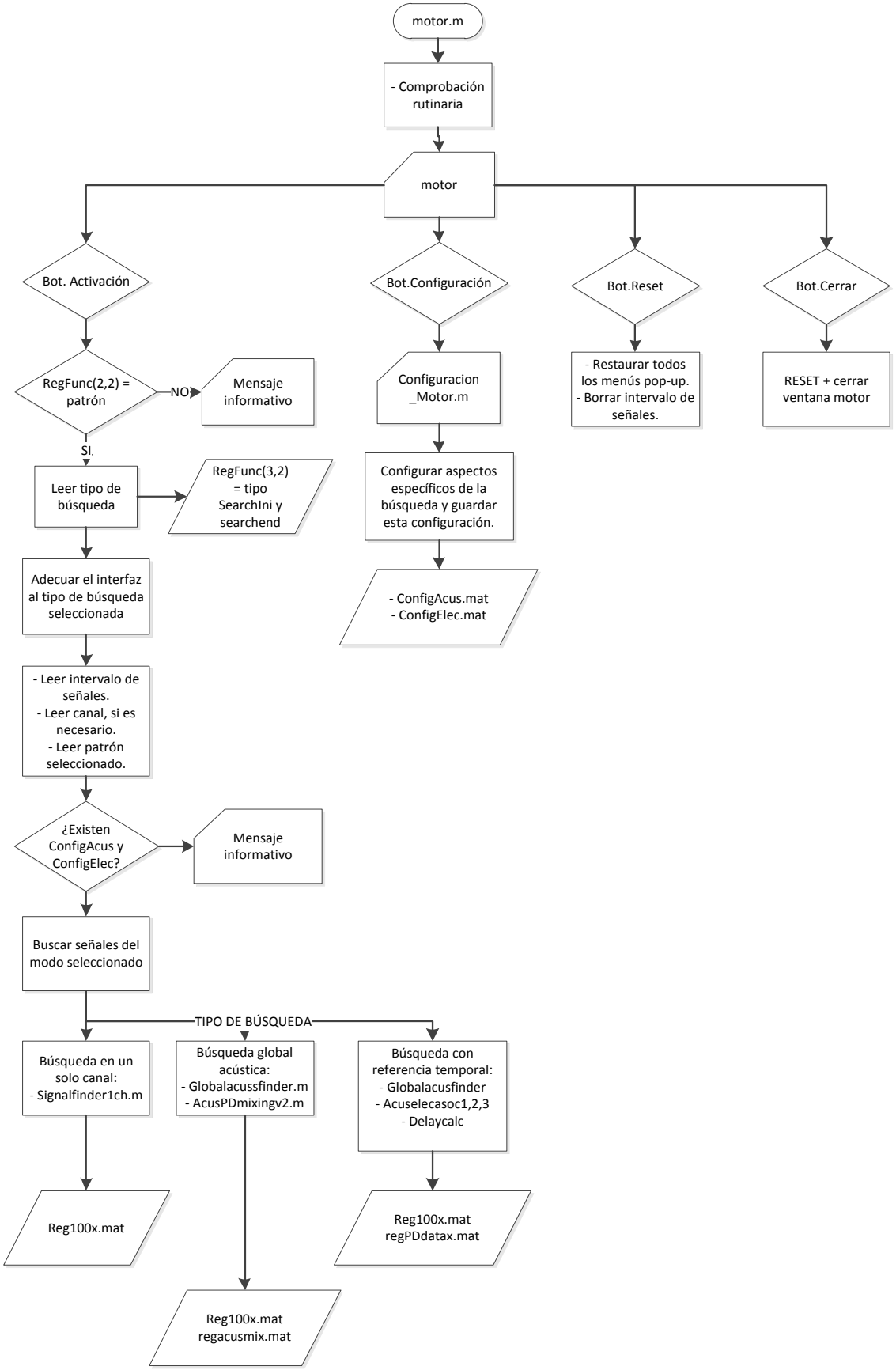


Figura 74.- Esquema de funcionamiento de la herramienta motor de búsqueda

- **Barra de progreso**

Se ha diseñado una función para poder informar al usuario de en qué estado se encuentra el programa. Para esta función nos basamos en la representación gráfica en objetos de tipo *axes.* Se ha optado por un display de 10 segmentos (Figura 75) para mostrar el proceso de la búsqueda.

La función creada responde al comando *ProgBar*, cuyos argumentos de entrada serán *ProgBar* (*handlepanel, num*) donde *handlepanel* será el puntero del panel donde queremos mostrar el *display* de 10 segmentos y *num* es el número de segmentos que se quieren mostrar.

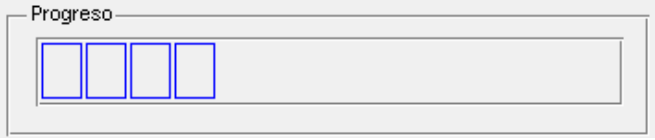


Figura 75.- Barra de progreso

Para ello se tendrá que conocer el proceso donde se aplicará esta función y subdividir este proceso en diez puntos importantes en el programa para poder distribuir de forma equitativa el tiempo de cada uno de los segmentos del *display* o simplemente saber que el programa está en determinado punto. Para cada punto importante, el programa pinta un cuadro más en la barra de proceso. Este cuadro se ha denominado *Barra.jpg.*

4.6.5 Desarrollo de la herramienta Visualización de eventos

Al pulsar el botón de Visualización de señales encontradas se abre la ventana correspondiente (Figura 76) a la cuarta herramienta o submenú *visualsignal.m*. Su funcionamiento se describe en la Figura 77.

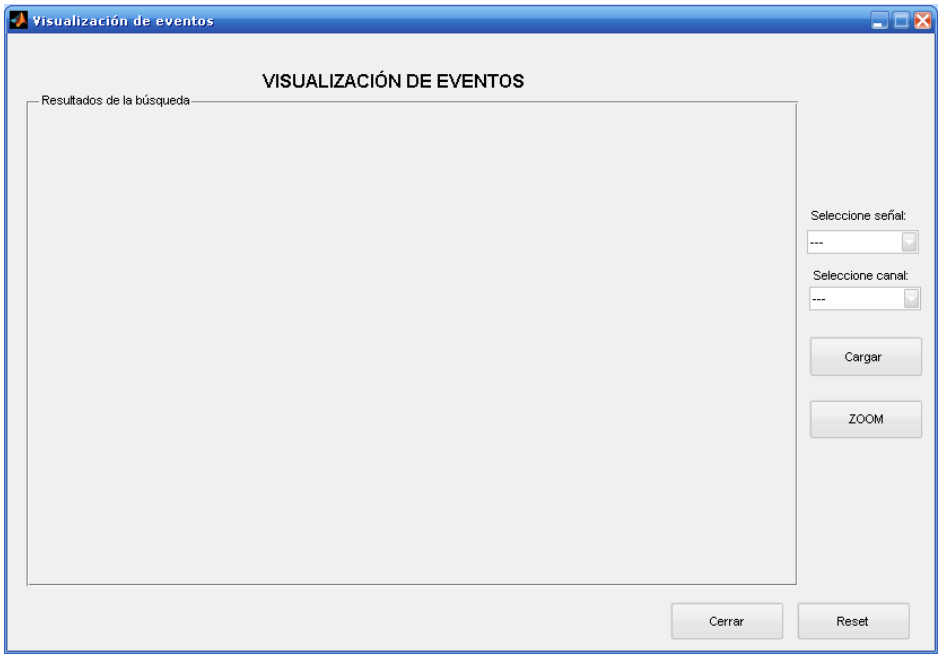


Figura 76.- Ventana diseñada para la herramienta de Visualización de los eventos encontrados

Al pulsar el botón Cargar el programa accede a los datos creados tras la búsqueda. De no existir búsqueda previa el programa informará al usuario de los pasos a seguir para el correcto funcionamiento del programa. En función de la búsqueda previa, el botón Cargar usará distintas funciones.

- **Búsqueda de 1 canal o global acústica:** Utiliza una modificación de la función *PlotSignalFind(filenamenum,reg100x,panel).* Esta función pinta las señales encontradas en los registros de búsqueda generados por estas señales sobre el panel principal de la aplicación. Los argumentos de esta función son los mismos que se describen en el capítulo 2 añadiendo el puntero al panel donde se va a pintar la gráfica (*panel*).

- **Búsqueda con referencia temporal:** Utiliza una modificación de la función *pintaPD(filenamenum,panel).* Esta función pinta los eventos considerados descargas parciales sobre sus señales originales en el panel principal de la aplicación. Los argumentos de esta función son la señal que se ha analizado (*filenamenum*) y el puntero del panel donde se va a pintar la gráfica (*panel*).

El botón de Zoom permite que seleccionando únicamente un intervalo en un solo canal, la gráfica entera se ajusta a esa selección. Para ello se hace uso de la función ZOOM que proporciona MATLAB.

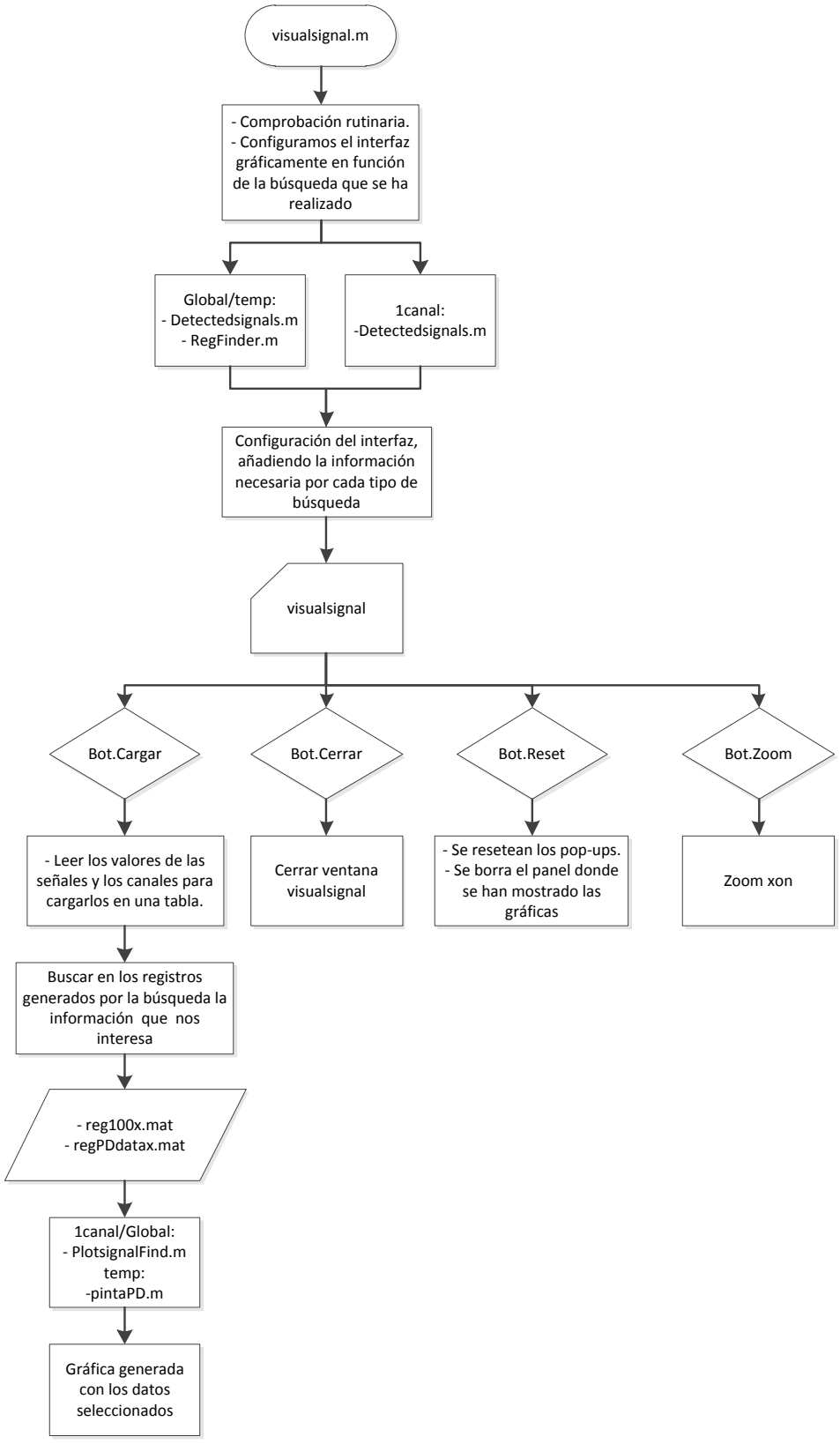


Figura 77.- Esquema de funcionamiento de la herramienta visualsignal

4.6.6 Desarrollo de la herramienta Visualización de la base de datos

Al pulsar el botón de Visualización de la base de datos de búsqueda se abre la ventana correspondiente (Figura 78) a la quinta herramienta o submenú *visualdata.m*. Su funcionamiento se describe en la Figura 79.

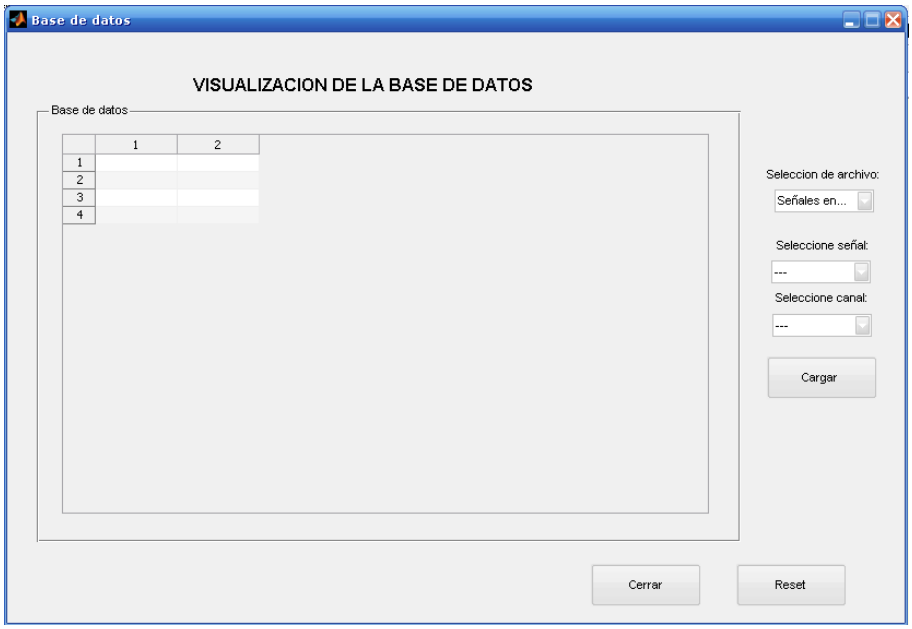


Figura 78.- Ventana diseñada para la herramienta de Visualización de la base de datos

Al crearse la ventana, el programa primero comprueba que tipo de búsqueda se ha realizado accediendo al archivo *RegFunc.mat*. En función del tipo de búsqueda realizada cargará unos determinados datos de las bases de datos creadas para la configuración de la pantalla. A continuación se describen las funciones utilizadas para cada tipo de búsqueda.

- **Búsqueda global acústica y búsqueda con referencia temporal:** Utiliza las funciones *Detectedsignals(canal)* y *RegFinder.m*. El argumento de entrada de la función *Detectedsignals* es el canal del cual se quiere saber el nombre de las señales donde se han encontrado eventos DP. La función accede a los registros generados en cada canal y lee el nombre de las señales que se han encontrado La función *RegFinder* comprueba los registros que existen y devuelve el nombre de estos registros.

- **Búsqueda en un solo canal**: únicamente hará uso de la función *Detectedsignals.m* descrita anteriormente.

Una vez configurada la ventana de visualización de la base de datos, pulsando el botón Cargar, se cargarán en la tabla los datos definidos en la configuración.

Si se ha realizado una búsqueda con referencia temporal, el programa ofrece la opción de cargar los datos de las señales encontradas en los canales acústicos o cargar los archivos

asociados a las descargas parciales encontradas (acústicas asociadas a algún evento eléctrico).

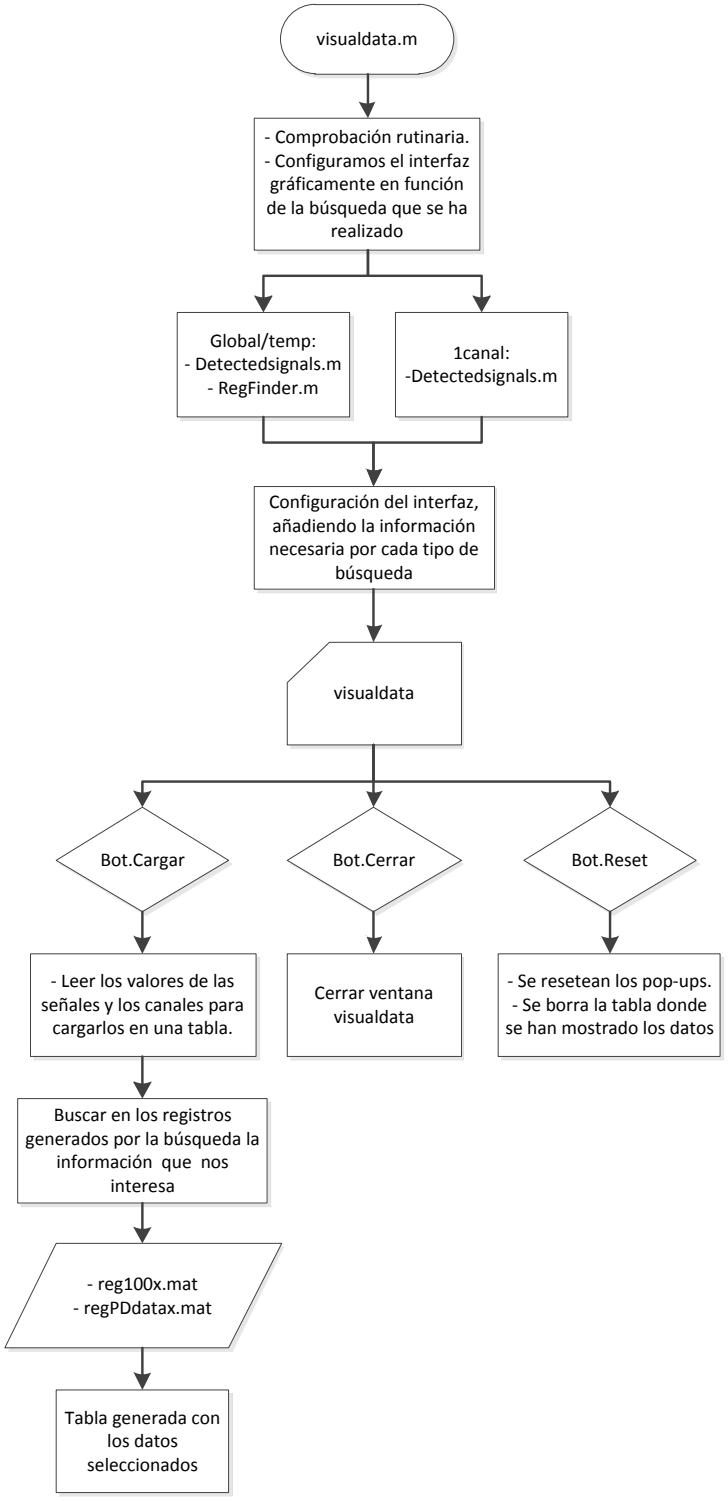


Figura 79.- Esquema de funcionamiento de la herramienta Visualización de la base de datos

4.6.7 Desarrollo de la herramienta Estadística

Al pulsar el botón de Herramienta estadística se abre la ventana correspondiente (Figura 80) a la sexta herramienta o submenú *estadistica.m*. Su funcionamiento se describe en la Figura 82.

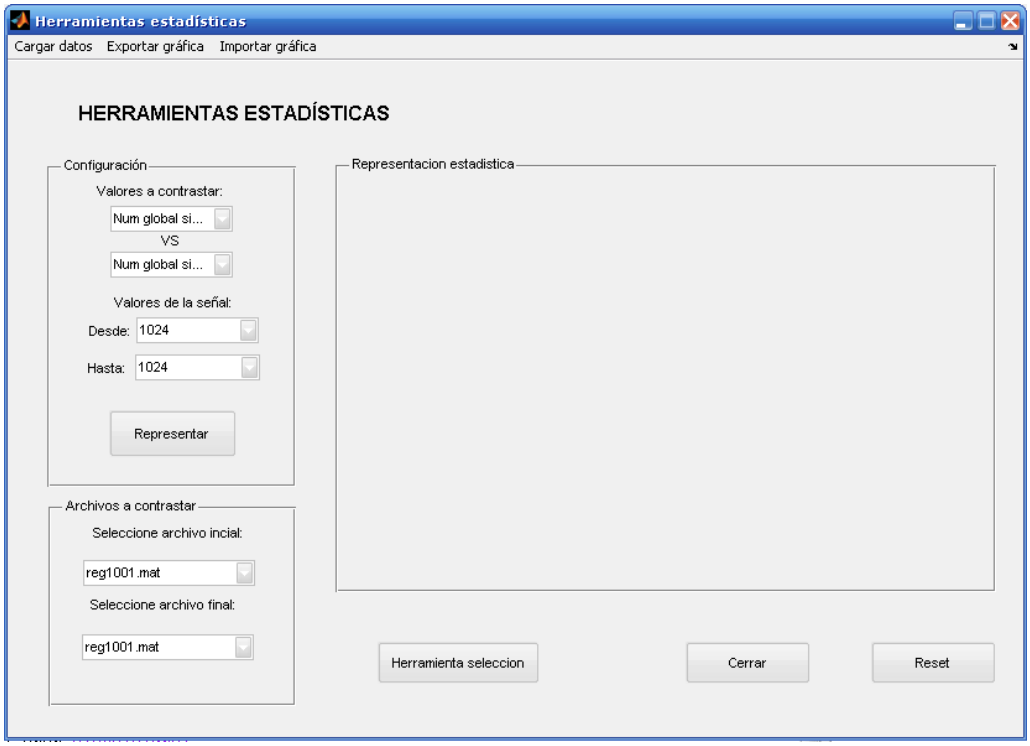


Figura 80.- Ventana diseñada para la herramienta estadística

Tras crearse la ventana, se realiza la misma configuración descrita en el apartado desarrollo de la Herramienta Visualización de Base de Datos para cargar los datos de las búsquedas en los menús de configuración. Para búsquedas de tipo global acústica y con referencia temporal el programa permite comparar los registros generados por cada canal, lo que permite comparar la información obtenida en cada canal.

Pulsando el botón Representa se pintarán los datos seleccionados en la configuración en una gráfica mediante la función *scatter* que representará los datos como puntos en la gráfica. Los datos se obtienen de las bases de datos generadas por las búsquedas. Los datos representados se almacenan en una variable llamada *dataest.mat.*

Pulsando el botón Herramienta selección, se pinta la misma gráfica en una nueva ventana con los mismos datos. Tras pintar esta gráfica se activa la herramienta *selectdata.m* que permite la selección de puntos. Estos datos se almacenan en la variable *selectest* (Figura 81) que contiene los siguientes datos:

selectest <28x2 cell>

	1	2
1	'Signals'	<1x2 cell>
2	'PDidx'	<25x1 doub...
3	'Signalw Am...	'Signalw Env...
4	0.1578	1.5492e-04
5	0.1706	1.7425e-04
6	0.1751	1.5402e-04
7	0.1574	1.5874e-04
8	0.2190	2.0916e-04
9	0.1921	2.3445e-04
10	0.2304	2.3168e-04
11	0.2099	1.9415e-04

Figura 81.- Variable *selectest* que contiene los datos seleccionados en la herramienta de selección estadísitca

- ***Signals*:** Nombre de las señales que se han representado y seleccionado con la herramienta de selección.

- ***PDidx*:** Índices asociados a los eventos DP correspondientes a las señales que se han seleccionado.

- ***selectest*(4,1) y *selectest*(4,2):** Nombre del tipo de datos que se representan el los ejes x e y respectivamente. Debajo de ellos se escriben los datos numéricos de la selección, cada fila correspondiendo a uno de los *PDidx* asociado.

Estos datos se podrá representar cargándolos desde el menú superior izquierdo de la ventana con el nombre Cargar datos.

Se ha añadido la posibilidad de exportar o importar las gráficas en formato JPG. Para ello, en los menús superiores de la pantalla, se selecciona el archivo correspondiente a la gráfica. Estos menús ejecutan las siguientes rutinas.

Para exportar las imágenes se ha usado el comando *print* de MATLAB. Este comando guarda objetos gráficos, en este caso una gráfica, en el formato que se le indique. Los argumentos de la función *print(h, format, filename)* son el puntero al objeto gráfico que se desea exportar (*h*), el nombre que se le quiere dar a ese archivo (*filename*) y el formato en el que se quiere guardar (*format*). Para exportar una imagen en formato JPG en el campo *format* habría que insertar el valor *-djpeg*.

Para importar una imagen únicamente hay que cargar el archivo que contiene la imagen con el comando *imread(filename)* siendo *filename* el nombre de la imagen que se desea cargar. Esa información que se ha cargado como una variable del programa se pasa como argumento a la función *image()*. En el caso de esta ventana, primero se hace un borrado del panel en el que se realiza el resto de representaciones para evitar que se superpongan datos.

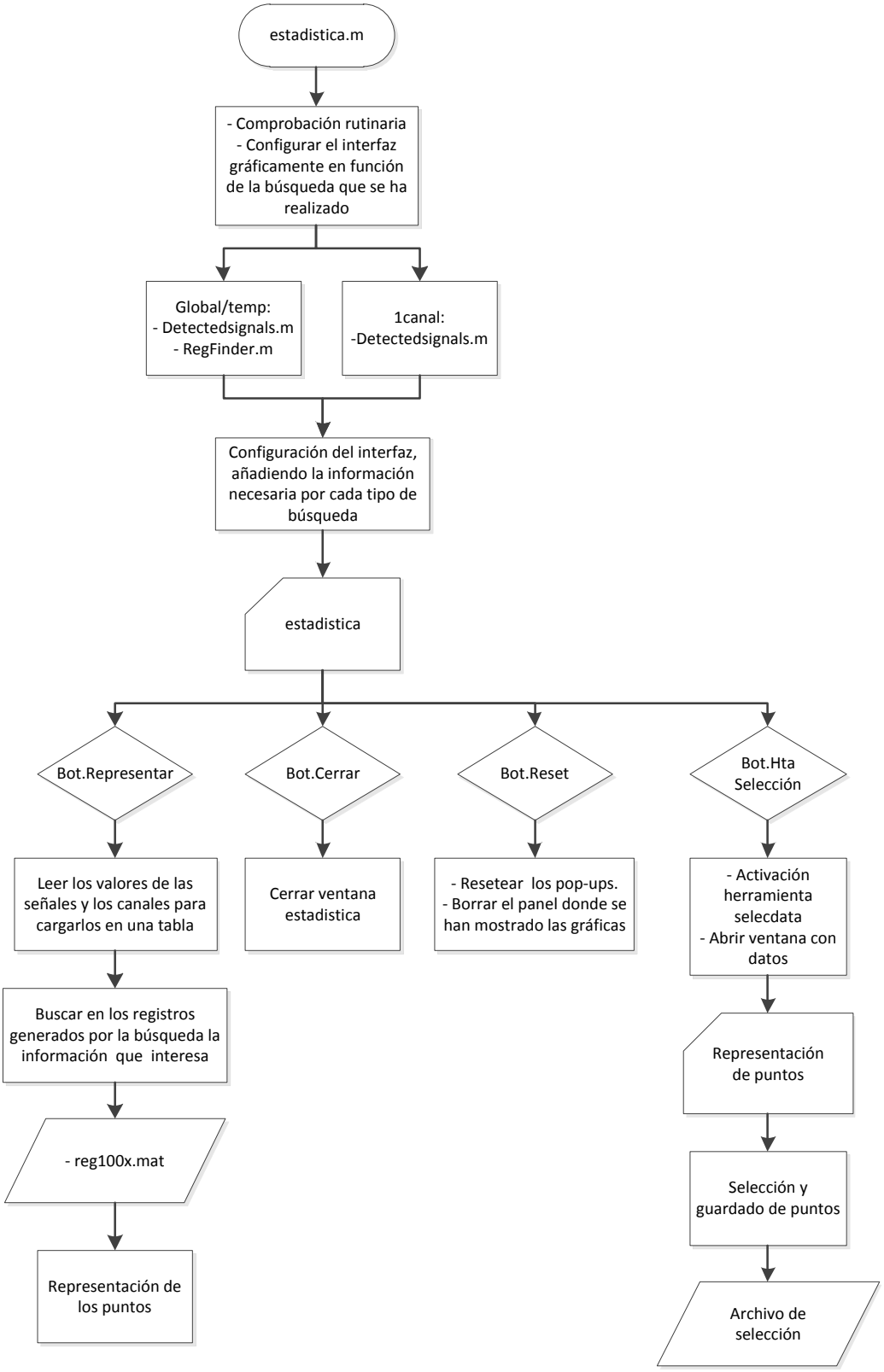


Figura 82.- Esquema de funcionamiento de la herramienta estadística

4.6.8 Desarrollo de la herramienta Localización

Al pulsar el botón de Visualización de la base de datos de búsqueda se abre la ventana correspondiente (Figura 83) a la séptima herramienta o submenú *local.m.* Su funcionamiento se describe en la Figura 84.

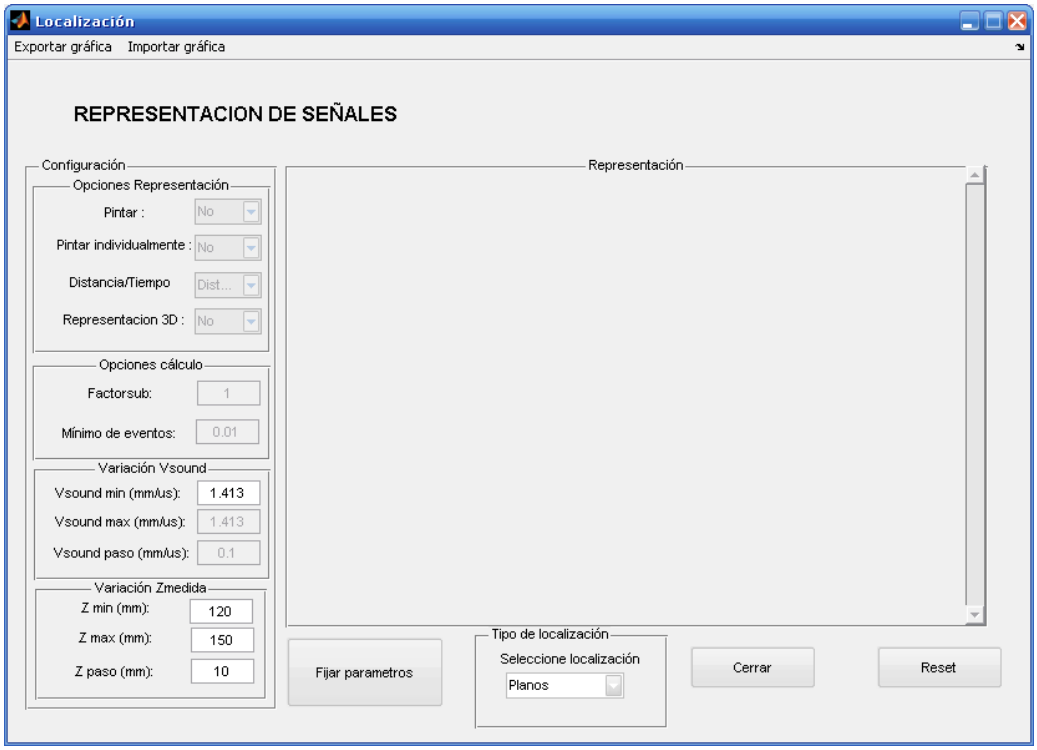


Figura 83.- Ventana diseñada para la herramienta de localización

Esta ventana también es auto configurable, por lo que en función de cada tipo de localización se podrán configurar únicamente ciertos valores. Los tipos de localización usan la función *local1.m,* que es la encargada de representar la localización espacial. Esta función es una modificación de las funciones *Lanzador2D.m* e *Histograma2D.m* para combinar su funcionalidad y poder representar todos los tipos de localización con una única función.

Para cada tipo de localización se ha definido una variable llamada *tipobusq* que adoptará un valor distinto para cada localización, definido por el menú Seleccione localización. Esta variable le indica a la función *local1.m* que tipo de localización realizar. En el capítulo 2 se detallan los distintos tipos de localización, que estarán asociadas a la variable *tipobusq* de la función *local1.m.*

Tras configurar la ventana se pulsa el botón Fijar parámetros. Este botón lanza la función *local1.m* con los valores de la configuración y el tipo de búsqueda para indicar a la función cual es la representación deseada.

Al igual que en el menú Herramienta estadística se ha añadido la posibilidad de exportar o importar las gráficas en formato JPG. Para ello se seleccionará el archivo correspondiente a la gráfica pinchando en los menús superiores de la pantalla. La forma de exportar e importar es la misma que en la herramienta anterior.

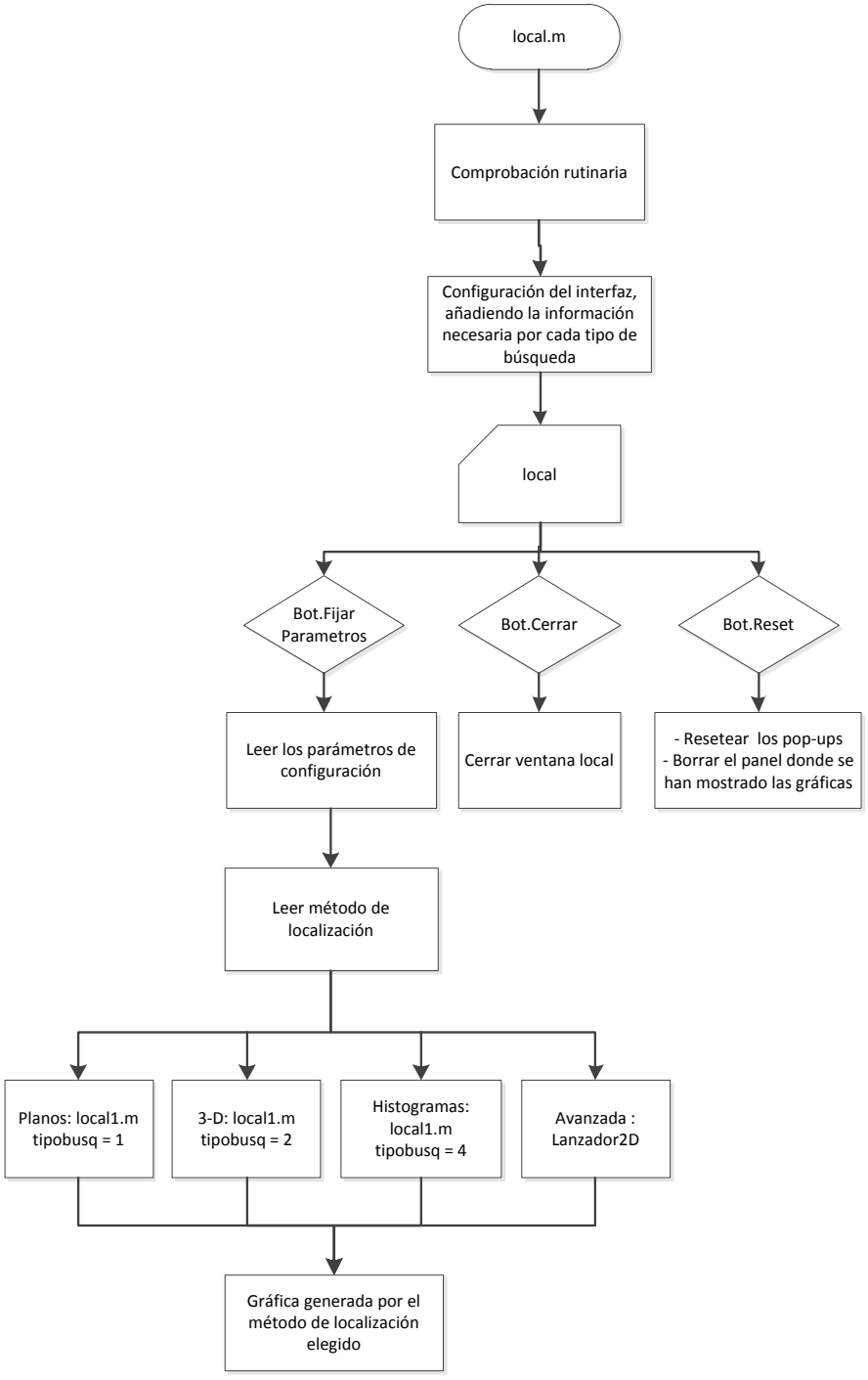


Figura 84.- Esquema de funcionamiento de la herramienta de localización

Tras crear todos los archivos tendremos la siguiente distribución de archivos. La Tabla 4 resume la estructura de archivos de la que se compone el programa final.

Tabla 4.- Tabla de las funciones desarrolladas para la gestión de la aplicación *PDtool*

Nombre del fichero dentro de /PDtool	Explicación del directorio
Leeme.txt	Explica el correcto uso del programa.
PDtool.m	Carga de la portada. Archivo de arranque
ToolLibrerias • Barra.jpg • Configuración_Motor.fig y .m • ContChanel.m • ContFile.m • ContSignal. • DetectedSignals.m • Escudo.jpg • Estadística.fig y .m • Histo2.m • Histograma2D.m • Hpatron.fig y .m • Imppatron.m • Lanzador2D.m • Local.fig y .m • Local1.m • Motor.fig y .m • Patrones.fig y .m • Pintaunpatron.m • Pintind2.m • Primera.fig y .m • Primaptronizar.m • Principal.fig y .m • ProgBar.m • RegFinder.m • Secpatronizar.m • *Selectdata*.m • Tercpatronizar.m • Visualdata.fig y .m • Visualsignal.fig y .m	Librerías creadas para la el diseño de la interfaz
Patrones	Contiene los patrones existentes.
Signals	Contiene las señales adquiridas.
Librerías	Contiene las librerías suministradas responsables de los cálculos

Capítulo 5

Resultados

Tras el diseño y depuración del código, se obtiene una primera versión estable (*PDtool v1.0*) que cumple con la funcionalidad requerida por el usuario experto. Para conseguir esta estabilidad se han tenido que afrontar numerosos casos en los que el programa entraba en algún estado que no era válido. Tras solucionar los errores detectados se obtiene esta primera versión.

Para estudiar su funcionalidad se probaron todos los casos posibles de detección y localización. Se utilizaron además distintos intervalos de señales. Se puede apreciar que el procesado de un alto nivel de señales requiere una gran cantidad de tiempo de análisis.

A continuación se muestra un ejemplo de uso acompañado de una serie de figuras en las que se ve el correcto funcionamiento de la herramienta para un caso genérico de detección y análisis de descargas parciales. Puede encontrarse mayor detalle en el anexo 1 (Manual del Usuario).

La primera ventana que se observa es la del menú principal, que contiene los botones a todos los submenús (Figura 85)

Figura 85.- Menú Principal

El primer paso es cargar un patrón, como se muestra en la Figura 86.

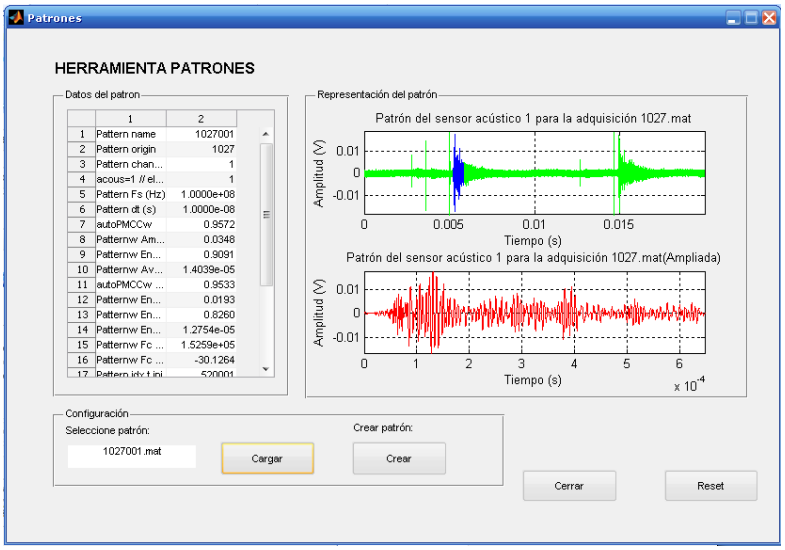


Figura 86.- Patrón cargado. Representación gráfica y tabla de datos

Si no se dispone de un patrón se puede crear como se muestra en la Figura 87.

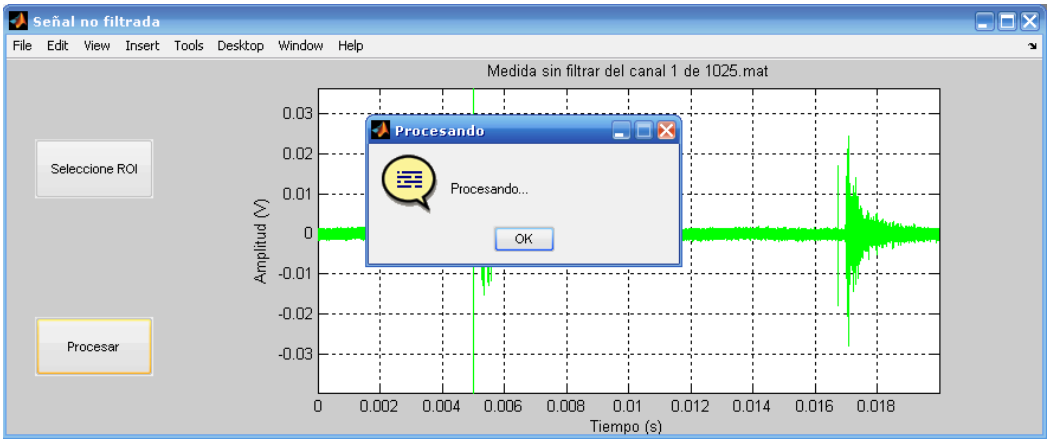


Figura 87.- Procesado de un patrón tras la selección de una parte de la señal

Observar las adquisiciones puede ayudar a la hora de seleccionar un canal del que extraer un patrón (Figura 88).

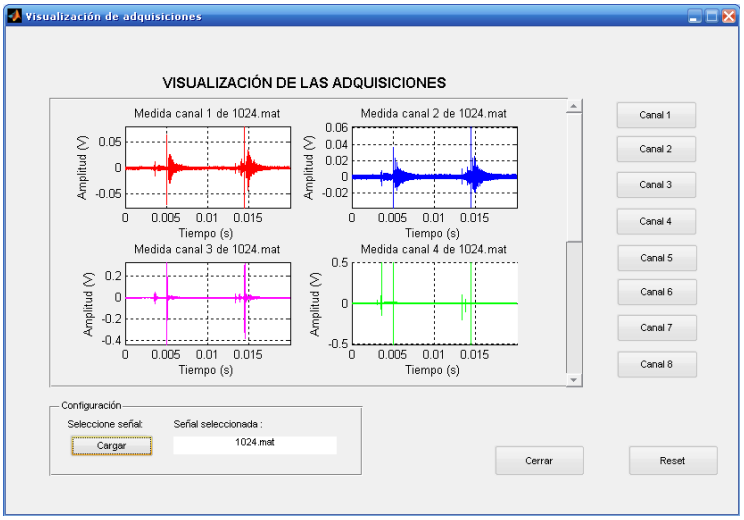


Figura 88.- Carga de una adquisición

Una vez se ha seleccionado el patrón, se pasa a la detección de descargas parciales. Para ello se configuran los datos básicos de la búsqueda como es el intervalo de señales en las que detectar descargas parciales y el tipo de detección que se va a emplear (Figura 89). También se puede realizar la configuración de otro tipo de datos más avanzados (Figura 90)

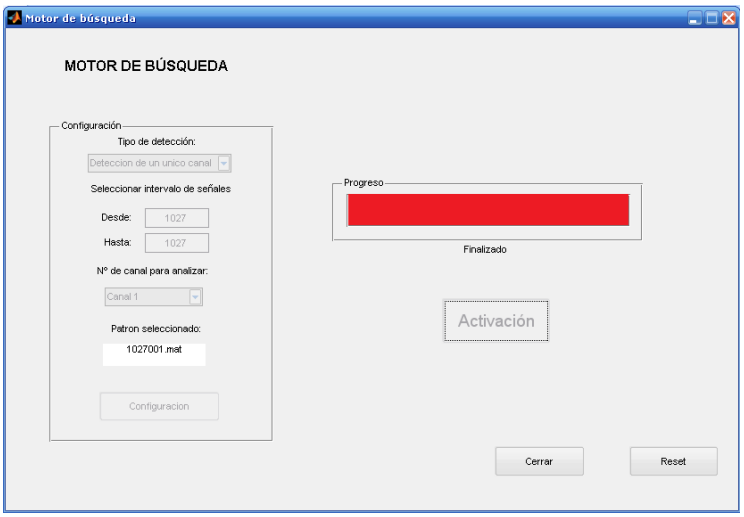


Figura 89.- Proceso de detección terminado

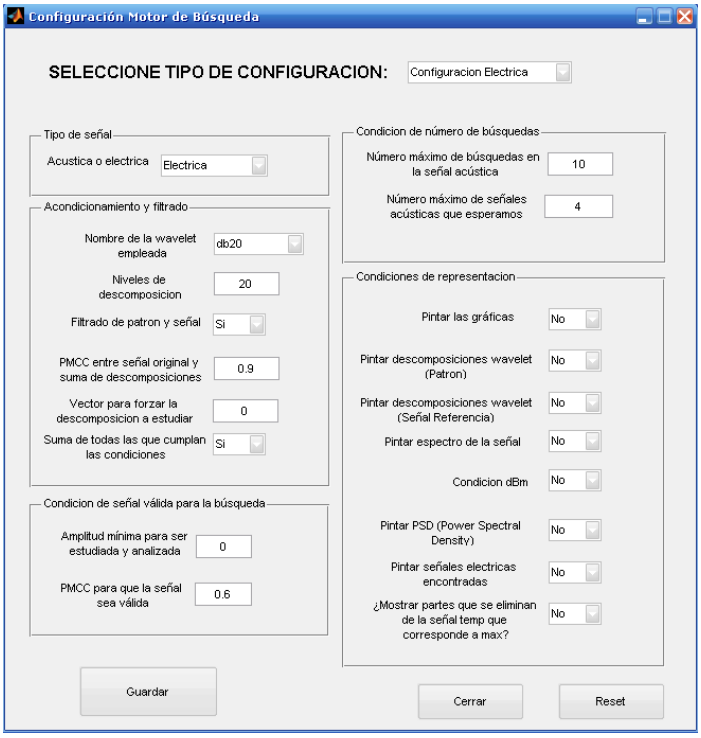


Figura 90.- Configuración de ciertos aspectos técnicos de las funciones de detección

Al finalizar la detección se puede empezar a ver los resultados de esta búsqueda. La primera herramienta es la visualización gráfica de eventos detectados. Esta muestra la información correspondiente a cada tipo de detección realizada previamente. En caso de ser una búsqueda simple o en el caso de ser múltiple, pero sólo acústica, la información a mostrar es la representada en la Figura 91

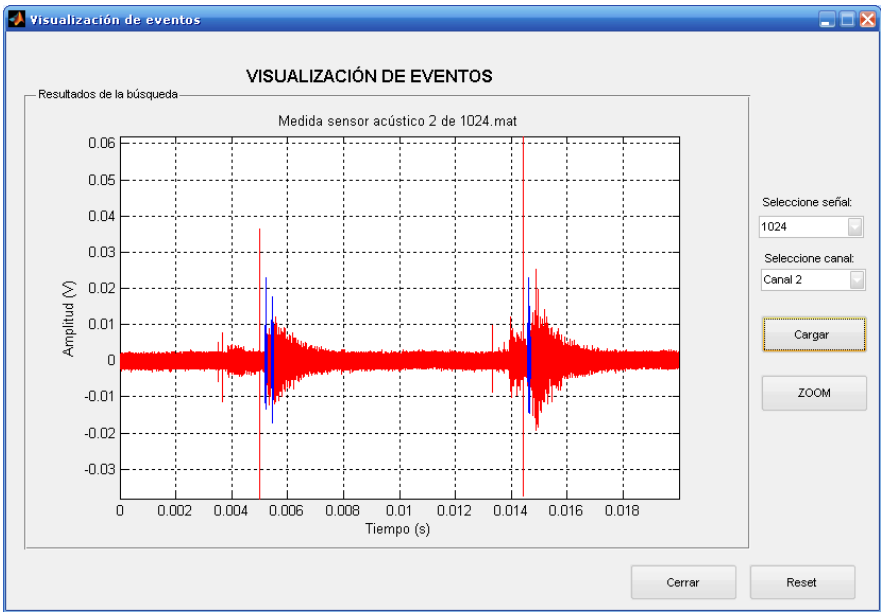


Figura 91.- Visualización de los eventos encontrados tras la detección

En caso de realizar una búsqueda con referencia temporal, se mostrarán todos los canales analizados y se podrá ver las señales de los eventos en cada uno de los cuatro canales (Figura 92).

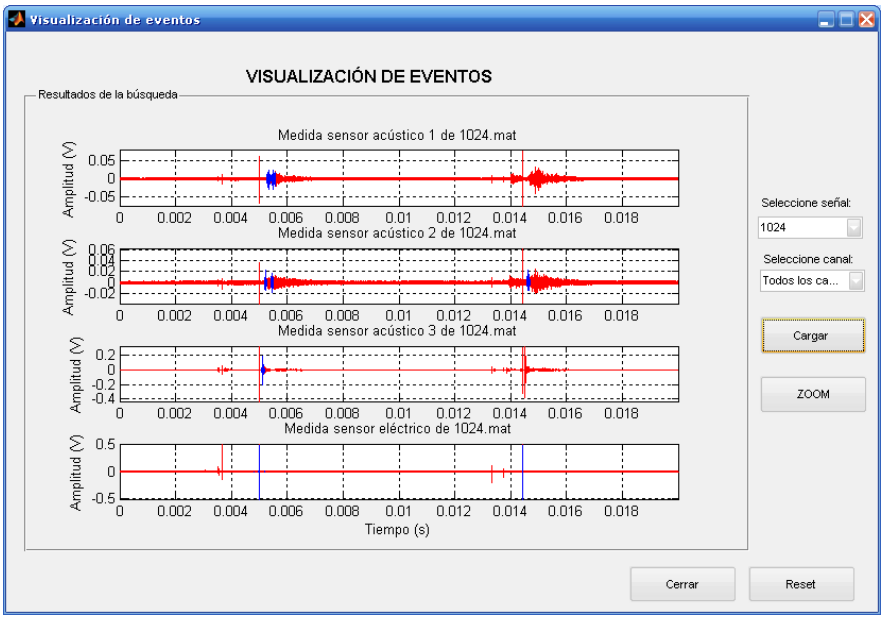


Figura 92.- Visualización de los evento PD válidos tras una detección con referencia temporal

La segunda herramienta es la visualización de las bases de datos generadas tras las detecciones. Estas mostrarán los datos numéricos (Figura 93) de cada evento detectado.

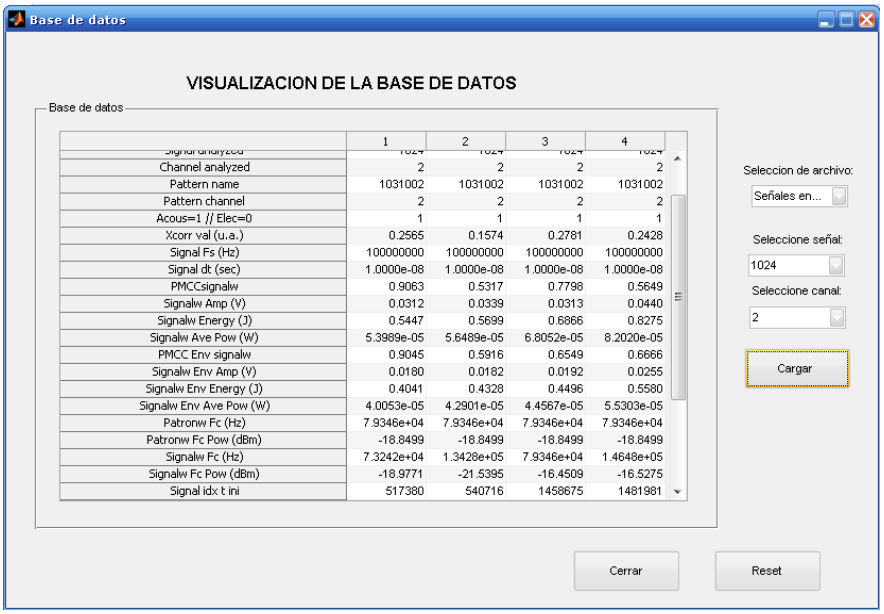


Figura 93.- Visualización de la base de datos de los eventos encontrados

La tercera herramienta (Figura 94) permite un análisis estadístico de las bases de datos generadas.

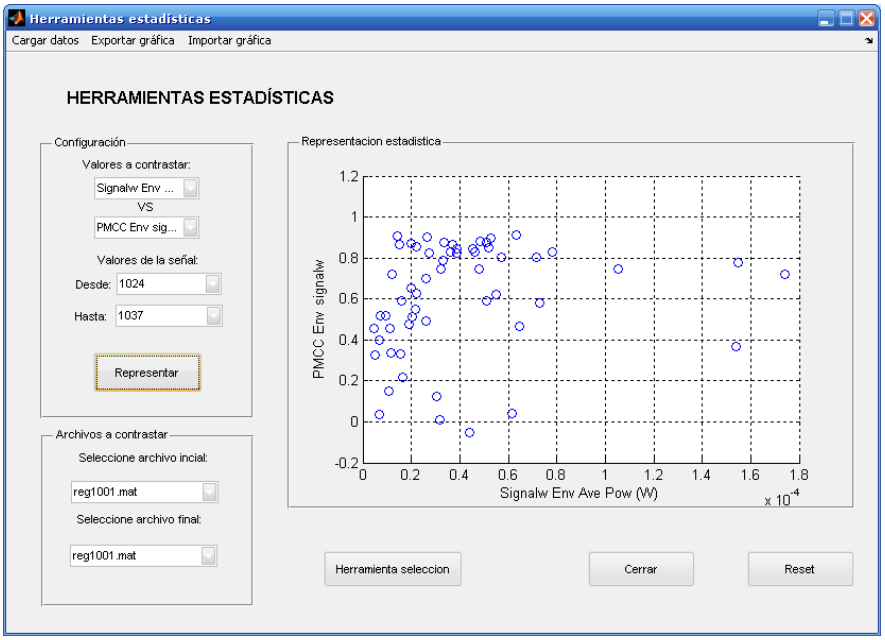


Figura 94.- Representación gráfica de los datos de los eventos encontrados en la herramienta estadística

La herramienta estadística permite la selección y almacenamiento de ciertos datos seleccionados de forma gráfica (Figura 95). Esta es otra forma de detección de descargas parciales.

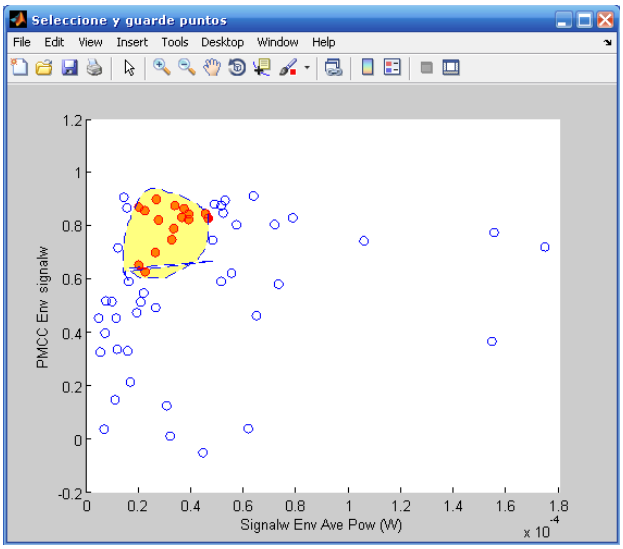


Figura 95.- Selección de los puntos de interés en la herramienta estadística

Por último está la localización espacial de las descargas parciales. Este menú permite configurar tres tipos básicos de localización: localización por planos (Figura 96), localización espacial tridimensional (Figura 97) y localización por histogramas (Figura 98).

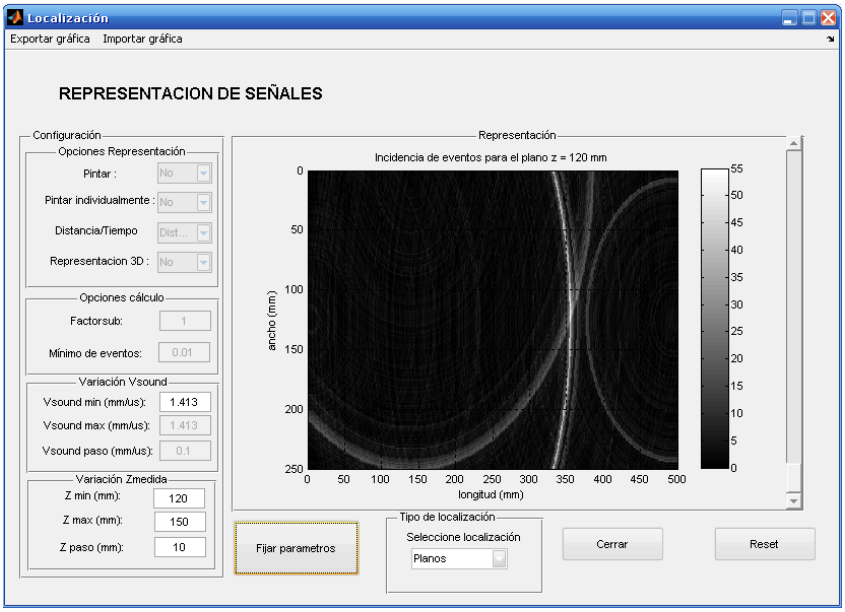


Figura 96.- Localización espacial por planos

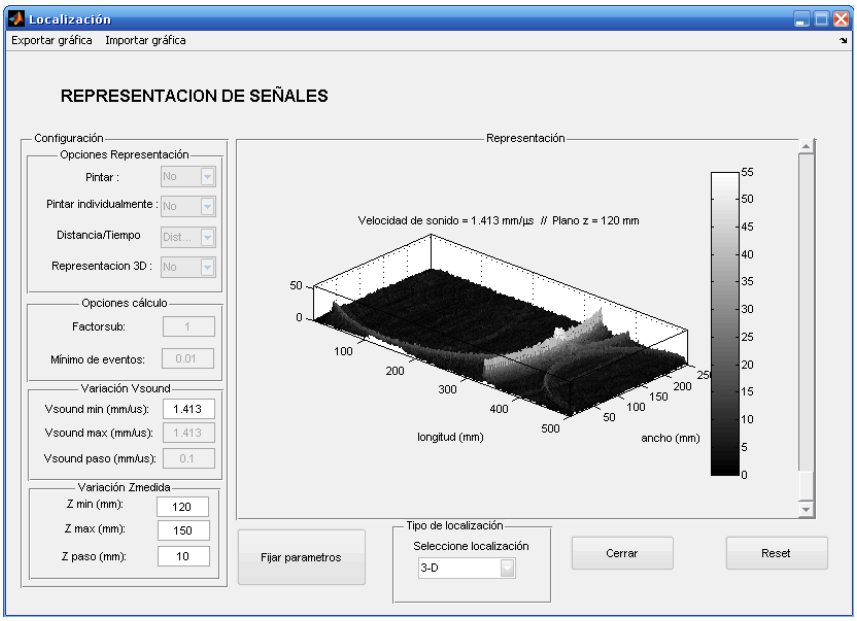


Figura 97.- Localización espacial tridimensional

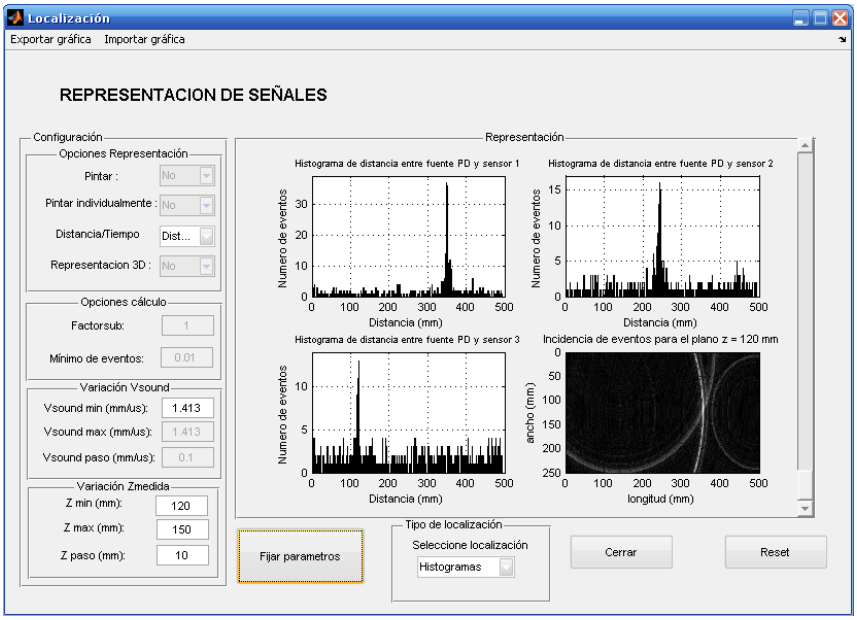


Figura 98.- Localización espacial con histogramas

Capítulo 6

Conclusiones y trabajos futuros

A continuación, se expone, de forma resumida un análisis sobre el cumplimiento de los objetivos propuestos y se presentan las principales conclusiones derivadas de los resultados a los que se ha llegado a lo largo de este proyecto. También, se enumeran las posibles líneas de trabajo que puedan dar continuidad al proyecto presentado.

6.1 Conclusiones

Se ha desarrollado una interfaz gráfica que permite al usuario procesar los datos obtenidos en experimentos de descargas parciales, pudiendo con ello facilitar su detección y análisis con medidas eléctricas y acústicas. Se han incorporado a esta aplicación todas las funciones para la detección y análisis de descargas parciales y los elementos de gestión necesarios de las bases de datos de descargas parciales.

La interfaz organiza secuencialmente las funciones de procesamiento de las señales. El usuario no necesita conocer las funciones que se ejecutan con la interfaz para realizar un diagnóstico ya que es el interfaz el que solicita la información y ejecuta las funciones.

La interfaz muestra los resultados obtenidos tras la detección y análisis de descargas parciales mediante gráficas y tablas.

La aplicación permite una configuración de todas las funciones utilizadas. Para ello se ha creado una configuración por defecto para que usuarios con formación específica puedan realizar el procesado.

La aplicación *PDtool* permite también realizar una configuración a nivel experto ya que se ha conservado la funcionalidad original del conjunto de funciones diseñadas por el usuario experto.

Se ha adjuntado un manual de usuario. Por ejemplo, el usuario puede consultar información sobre cuáles deben ser los pasos que debe seguir en cada pantalla, información sobre los gráficos obtenidos, etc.

La aplicación se basa únicamente en un sistema de ventanas que muestra la información asociada a cada parte del proceso de detección y análisis de descargas parciales. Se ha creado un sistema de ventanas que minimiza el número de ellas que se despliega.

La interfaz tiene una estructura clara y ordenada. Un usuario poco familiarizado con la detección y análisis de descargas parciales aunque con formación en el procedimiento puede obtener información suficiente para extraer unas primeras conclusiones. Otro factor a resaltar es el desarrollo de *PDtool* de forma guiada, de tal modo que el usuario no se pueda perder o dudar sobre cuál es el procedimiento a seguir. Este objetivo se ha conseguido incluyendo en la interfaz mensajes de error o de información que no permiten avanzar al usuario si no ha introducido toda la información necesaria en cada pantalla, o si no se ha introducido de forma correcta.

Se ha desarrollado también un sistema de selección de datos que cumple con la funcionalidad requerida por el usuario en las herramientas correspondientes a la creación de patrones y de análisis estadístico.

El desarrollo de la aplicación, a nivel de programación, se ha realizado de forma clara facilitando a desarrolladores futuros ampliar la aplicación en versiones futuras.

6.2 Trabajos futuros

Como línea de trabajo futuro se puede plantear una aplicación con la misma funcionalidad que esta, pero que realice la detección y localización en tiempo real (Figura 99). En este caso la interfaz también sería capaz de controlar la adquisición de datos, permitiendo por ejemplo variar la velocidad de muestreo u otros aspectos de la adquisición.

Figura 99.- Aplicación futura en la que la interfaz interactúa con la adquisición, procesando la información en tiempo real

Este trabajo se podría realizar en MATLAB ya que se dispone de las librerías para el procesado de la información y también permite la conexión con equipos de instrumentación. Sin embargo, otra opción sería desarrollarlo en otro entorno como puede ser LabVIEW, traduciendo el código de lenguaje MATLAB. LabVIEW es una excelente herramienta para crear interfaces en línea con equipos de medida. Además facilita la integración de programas de MATLAB (*scripts*) y proporciona bloques de funciones en instrumentación virtual que pueden sustituir total o parcialmente el programa de gran parte de las funciones implementadas en MATLAB.

También es posible integrar nuevos submenús en caso de ser necesaria otra herramienta para un nuevo aspecto del procesado. Gracias al tipo de programación que se ha seguido a la hora de desarrollar la aplicación es fácilmente adaptable a cambios en posibles versiones 2.X de la herramienta.

Capítulo 7

Presupuesto

7.1 Desglose y presupuesto total

Este documento contempla el coste total del diseño de la herramienta *PDtool* así como un pequeño desglose de los conceptos económicos.

Para el diseño de esta herramienta se ha tenido que realizar un estudio previo del experimento y de las funciones suministradas. Tras el estudio se procedió a realizar unas entrevistas con el usuario. Por último se procedió a realizar el diseño de la interfaz. Todo esto supuso un tiempo total de 6 meses (Tabla 5).

Como concepto de materiales empleados (Tabla 6) cabe citar el equipo empleado en el diseño de la interfaz como son el equipo portátil LG E500 valorado en 422 € y una licencia del programa MATLAB R2009a valorada en 1950 €.

Tabla 5.- Presupuesto de personal

Nombre y Apellidos	Categoría	Dedicación (hombres-mes)[2]	Coste hombre-mes	Coste (EURO)
Pablo Grandas Aguado	Ingeniero Técnico Industrial	6	2.694,39	16.166,34
				16.166,34

Tabla 6.- Presupuesto de equipo

Descripción	Coste (EURO)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable (EURO)
MATLAB R2009a	1.950	100	6	60	195
Portátil LG E500	422	100	6	60	42,2
					237,2

Tabla 7.- Presupuesto total

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	16.166,34
Amortización	237,2
Costes Indirectos	3.281
Total	**19.684**

El presupuesto total de este proyecto asciende a la cantidad de DIECIENUEVE MIL SEISCIENTOS OCHENTA Y CUATRO EUROS.

Leganés a 18 de noviembre de 2011

El ingeniero proyectista

Fdo. Pablo Grandas Aguado

[2] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)

Referencias

[1]. **IEC 60270.** "High-voltage test techniques. Partial discharge measurements. IEC 60270". International Electronic commission, 2000.

[2]. **Rubio-Serrano, Jesús, Posada, Julio E. and García Souto, José A.**, "Procesamiento para detección, identificación y localización de señales acústicas y eléctricas provenientes de descargas parciales." SAAEI 2010.

[3]. **Rubio-Serrano, Jesús.** "UC3M : INFORME SOBRE LOCALIZACIÓN ACÚSTICA." 2010.

[4]. **The Mathworks Inc.** "Creating Graphical User Interfaces (MATLAB©)." [Online] [Consultado: 13 de Noviembre de 2010.] http://www.mathworks.com/help/techdoc/learn_matlab/f5-998197.html.

[5]. **Guerrero Barragán, Diego Orlando.** "MANUAL DE GUI EN MATLAB." [Online] [Consultado: 2 de Febrero de 2011.] http://es.scribd.com/doc/15532859/MANUAL-DE-GUI-EN-MATLAB.

[6]. **The Mathworks Inc.** "41 Complete GUI examples." [Online] Julio 27, 2009.[Consultado: 2 de Febrero de 2011.] http://www.mathworks.com/matlabcentral/fileexchange/24861-41-complete-gui-examples.

[7]. **Sanz Arranz, Álvaro. "**Diseño de una herramienta con MATLAB para la adquisición y procesamiento de señales. Aplicación a sistema de detección de fallos de rodamientos." Proyecto de Fin de Carrera de la Universidad Carlos III de Madrid Octubre 2010.

[8]. **D'Errico, John.** "Herramienta *selectdata*." [Online] Febrero 19, 2007. [Consultado: 2 de Febrero de 2011.] http://www.mathworks.com/matlabcentral/fileexchange/authors/679.

[9]. **Mayhew, D.** "The usability engineering lifecycle: a practitioner's handbook for user interface design", Ed. , 1999.

[10]. **OCW UC3M.** "Curso de Interfaces de Usuario de Ingeniería Informática.". Open Course Ware de la Universidad Carlos III de Madrid (OCW-UC3M) [Online] [Consultado: 3 de Marzo de 2011.] http://ocw.uc3m.es/ingenieria-informatica/interfaces-de-usuario

Índice alfabético

Anexo 1: Manual de usuario

10.1 Introducción

El objetivo de este manual es explicar el funcionamiento del programa, así como las posibilidades que este proporciona. Para ello se explicarán todos los modos de funcionamiento.

10.2 Instalación e inicio

La herramienta no requiere de una instalación como tal pero es necesario contener todos los archivos y carpetas suministrados, tal y como se muestra en la Figura 100, en una única carpeta. De no ser así el programa no funcionará correctamente.

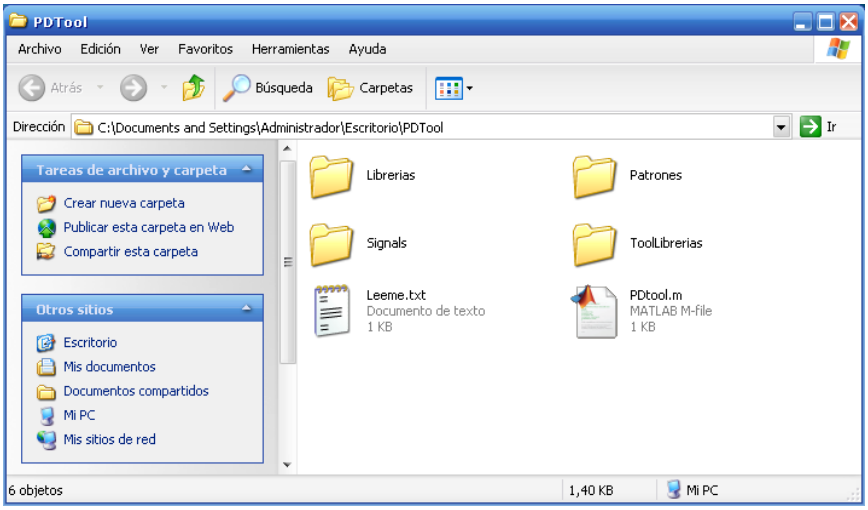


Figura 100.- Carpeta *PDtool* descomprimida

Para ejecutar la aplicación desde MATLAB se debe escribir en la línea de comandos *PDtool*. Este comando abre la ventana de presentación (Figura 101), donde se muestra la versión del programa así como otra información y donde se elige el idioma en el que se quiere trabajar. Una vez elegido idioma, la ventana se cerrará. En caso de haber alguna

búsqueda previa guardada, el programa lo indicará con un mensaje de información. Tras este mensaje se abre el Menú Principal donde se empieza a trabajar con el programa.

Figura 101.- Portada del programa

10.3 Menú Principal

Figura 102.- Menú Principal

La Figura 102 muestra la pantalla del Menú Principal en español. En esta ventana se observan distintos botones que abrirán cada uno la pantalla correspondiente a cada herramienta. El Menú Principal permanecerá visible en todo momento, aunque se esté trabajando en otra ventana.

Si no hay ninguna búsqueda previa, la evolución del programa guiará al usuario a abrir la ventana de *Herramienta patrones* para seleccionar un patrón para la búsqueda a través de mensajes informativos. Si se intenta entrar en ventanas de procesos posteriores a la selección del patrón, el propio programa le informará de los pasos a seguir para poder realizar una búsqueda.

En caso de haber una información previa guardada se podrá acceder a cualquiera de las 4 últimas ventanas (*Visualización de señales encontradas, Visualización de base de datos, Herramientas estadísticas* o *Localización*), que son las que muestran los resultados de la búsqueda.

10.4 Herramienta patrones

El primer paso para el análisis de DP es seleccionar un patrón representativo para el análisis. Con esta herramienta se puede cargar patrones previamente guardados o crear nuestros propios patrones de una forma muy gráfica.

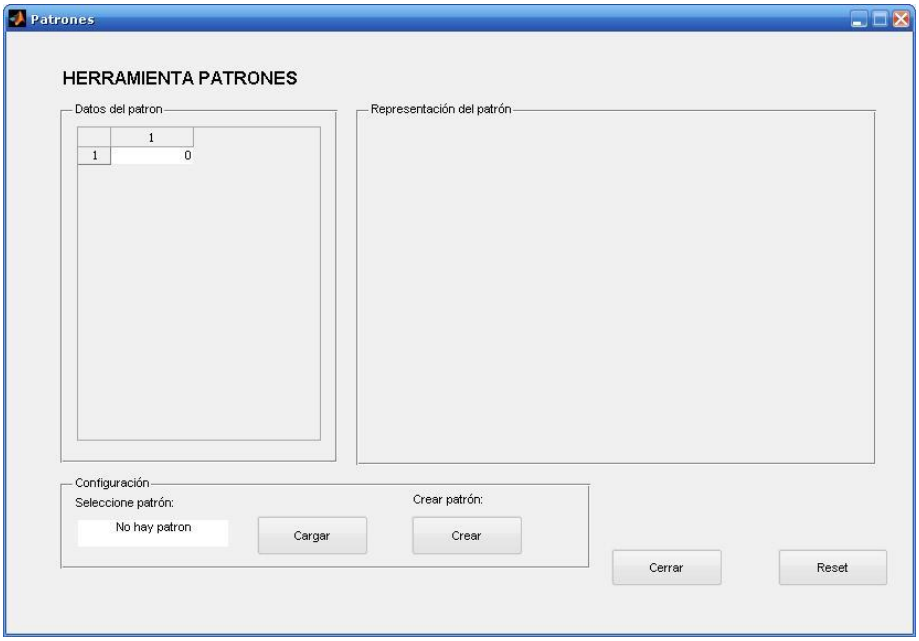


Figura 103.- Ventana de la herramienta Patrones

Pulsando en el Menú Principal el botón *Herramienta patrones* se abrirá la pantalla Herramienta patrones donde se observan los siguientes objetos (Figura 103):

- Tabla *Datos del patrón:* En esta tabla se mostrarán los datos numéricos del patrón.

- Panel *Representación del patrón*: En este panel se mostrarán dos gráficas. En la parte superior se mostrará la señal original que contiene el patrón, así como la porción de dicha señale que se seleccionó para el propio patrón. En la parte inferior únicamente el patrón ampliado.

- Botones *Cargar, Crear, Cerrar* y *Reset*: Los botones Cerrar y *Reset* cumplirán las mismas funciones en todas las ventanas. Cerrar cerrará la ventana y *Reset* pondrá la ventana en su valor por defecto, eliminando las gráficas y todos los datos creados en esta pantalla. El botón *Cargar* abre la ventana (Figura 104) de búsqueda de archivos. Seleccionar el que interese y presionar *Abrir*. Si se intenta cargar un archivo que no sea de tipo *patrón* el programa avisará de que el archivo no es correcto. El botón Crear abre la ventana Crear patrón que se describirá en el apartado siguiente (10.4.1) (Figura 106).

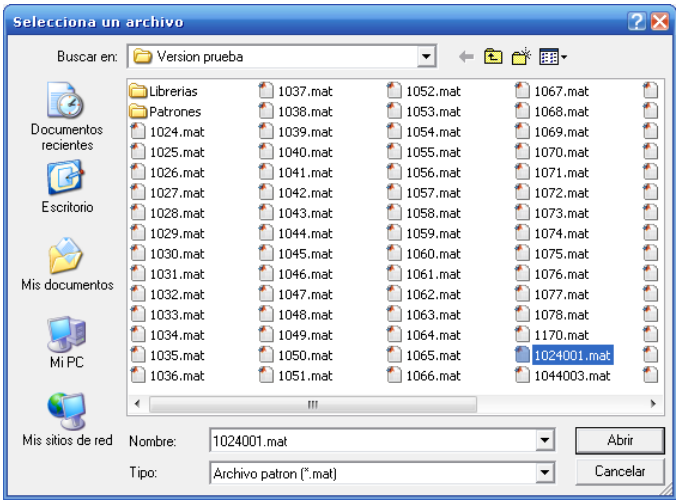


Figura 104.- Carga de archivos de tipo "patrón"

Tras cargar un patrón, se mostrarán los datos de este como se muestra en la siguiente figura (Figura 105).

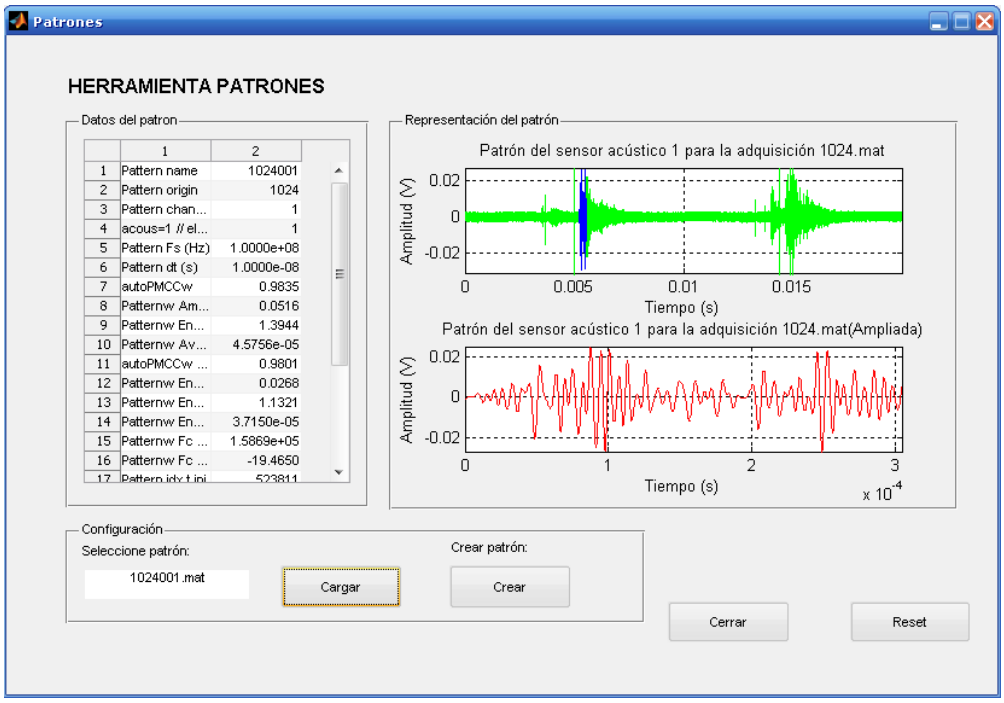


Figura 105.- Patrón cargado

Si no se dispone de patrones para cargar o se desea crear algún patrón nuevo, pulsando Crear aparecerá la ventana Crear Patrón (Figura 106).

10.4.1 Crear patrón

Figura 106.- Herramienta Crear patrón

Con la herramienta Crear Patrón se puede crear un patrón a partir de una señal. Como se observa en la Figura 106, la herramienta Crear Patrón dispone de un botón para cargar la señal, un menú desplegable para seleccionar el canal a mostrar, otro menú desplegable para definir el tipo de señal y un botón para representar la configuración elegida. Como todas las ventanas también dispone de un botón para cerrar la ventana y otro para reiniciarla.

Una vez seleccionados todos los parámetros de la señal que se desea para crear el patrón, pulsando el botón Representa se crea una nueva ventana (Figura 107) con una gráfica donde se muestra la señal y dos botones para su procesado.

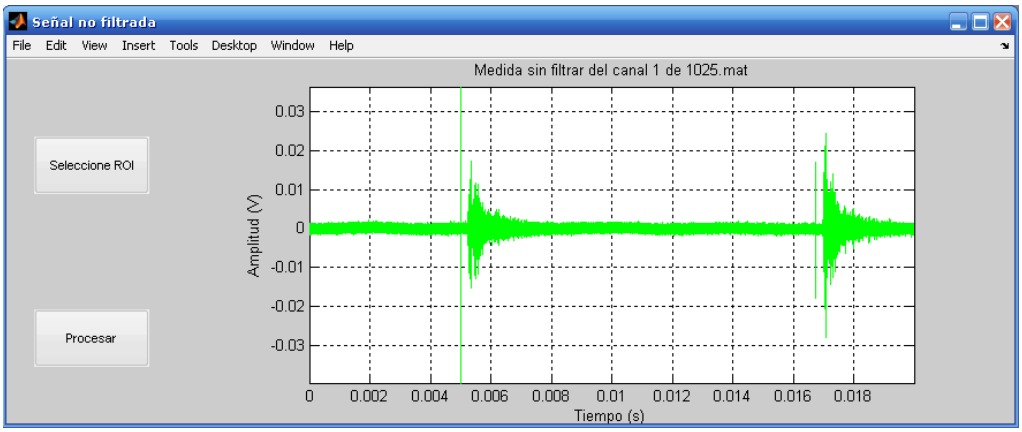


Figura 107.- Señal seleccionada no filtrada

Los pasos a seguir para la creación son los siguientes:

- Seleccionar región de interés (ROI).
- Primer procesado de la señal. En este procesado se filtra la señal para una mejor.

El botón 'Seleccione ROI' habilita la herramienta de selección para seleccionar una región de interés de nuestra señal (Figura 108) que se considere válida como patrón. Al pulsar el segundo botón 'Procesar', se hará un zoom a la selección de la señal y un filtrado de esta para que la señal en la que se basa el patrón sea lo más limpia posible (Figura 109). Aparecerá una ventana informativa que indica que se está realizando el filtrado.

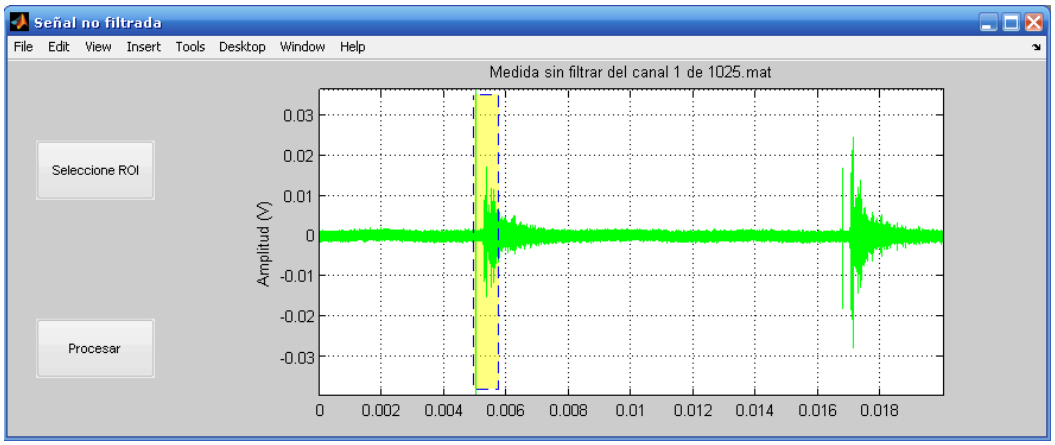


Figura 108.- Selección de un patrón dentro de la señal seleccionada

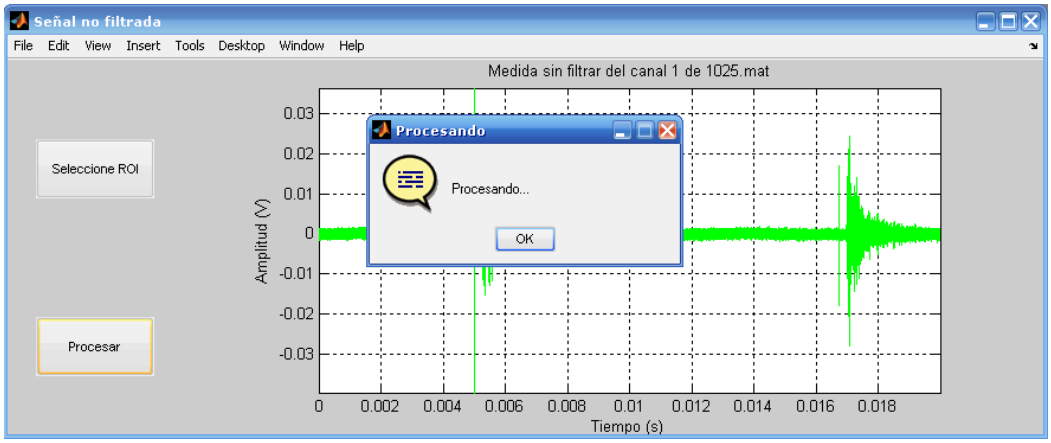


Figura 109.- Primer procesado del patrón

Tras el filtrado, se abrirá una nueva ventana (Figura 110) con la señal filtrada en color verde y la envolvente de la señal en color morado. También se creará un botón para realizar el procesado final del patrón. Si la selección y el filtrado son correctos, pinchando en el botón Procesar Patrón se creará el patrón.

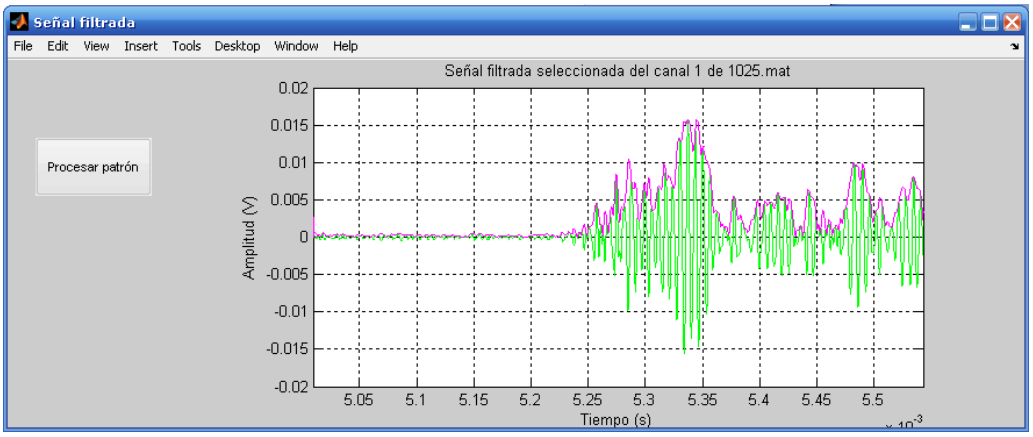


Figura 110.- Señal del patrón filtrada

Una vez termine la computadora de realizar los cálculos pertinentes, cerrará ambas ventas (Figura 107 y Figura 110) y abrirá una última ventana (Figura 111) donde se mostrarán la señal original con el patrón sobrepuesto. El programa indicará que se ha terminado de crear y patrón y mostrará el nombre de dicho patrón. Este patrón estará listo para utilizar.

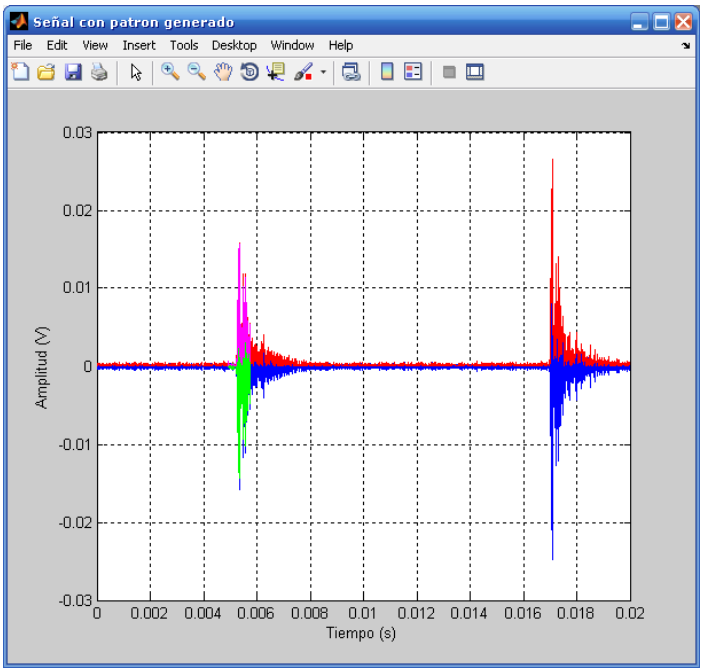


Figura 111.- Patrón final sobre su señal original

10.5 Visualización de las adquisiciones

La ventana creada en el submenú Visualización de las adquisiciones se usa para visualizar todos los canales de una señal adquirida. Se ha diseñado para que sea capaz de visualizar señales de hasta ocho canales. La ventana (Figura 112) contiene un panel central en el que se mostrarán las señales de cuatro en cuatro por medio de una *slidebar* o *slider* (Figura 113 y Figura 114). La herramienta representará los cuatro primeros canales de la señal y al pulsar el *slider* se representará los siguientes cuatro últimos canales, en caso de existir.

Contiene los botones de cerrar y *Reset* como el resto de ventanas y su función es la misma, cerrar la ventana (‘Cerrar’) y borrar gráficas, variables creadas e indicadores (‘*Reset*’).

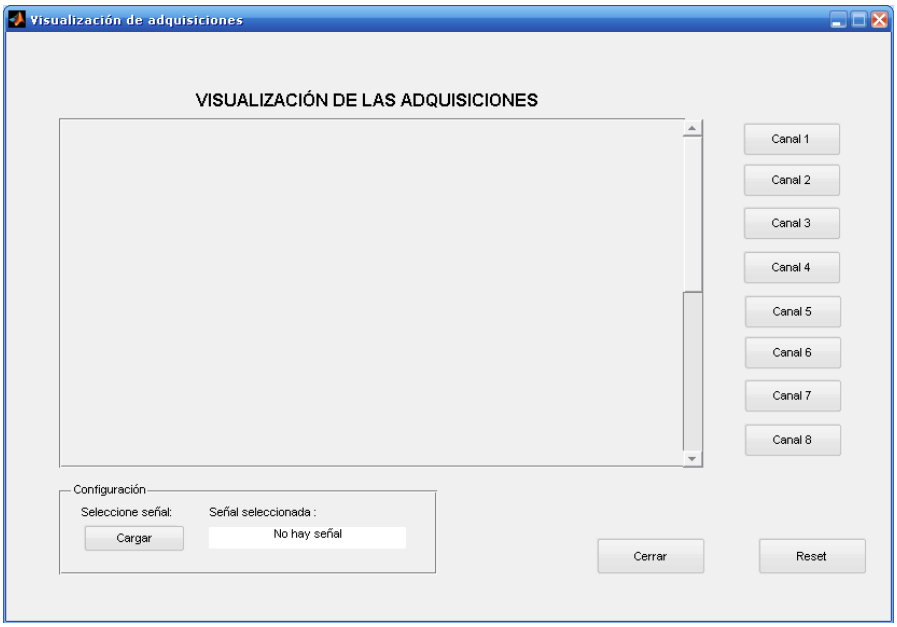


Figura 112.- Ventana de la herramienta Visualización de adquisiciones

También tiene un botón de carga, para cargar el archivo de la señal. El procedimiento de carga es idéntico al del patrón. Se abre una ventana con los archivos que hay en el directorio raíz del programa. El programa comprobará que el archivo es de tipo 'señal' y si es así lo cargará.

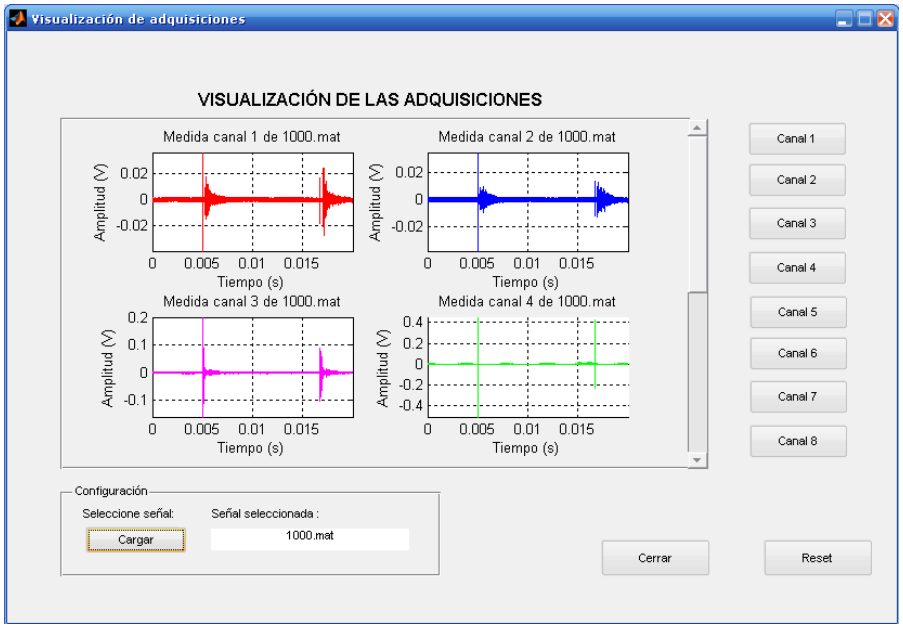


Figura 113.- Adquisición cargada (se muestra del canal 1 al 4)

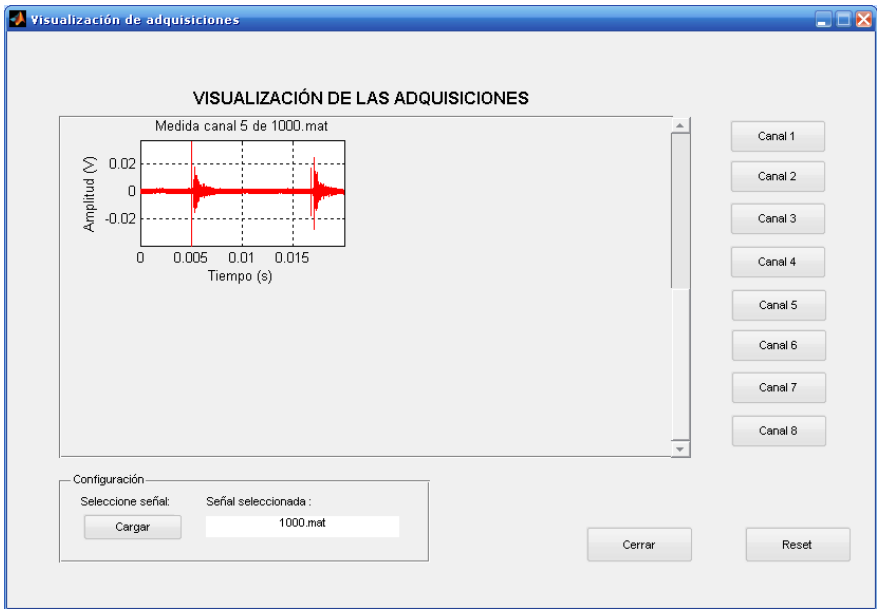


Figura 114.- Adquisición cargada (se muestra del canal 5 al 8)

En la parte derecha de la pantalla 'Visualización adquisiciones' se pueden ver ocho botones, que mostrarán individualmente (Figura 115) el canal que se desee, si este existe.

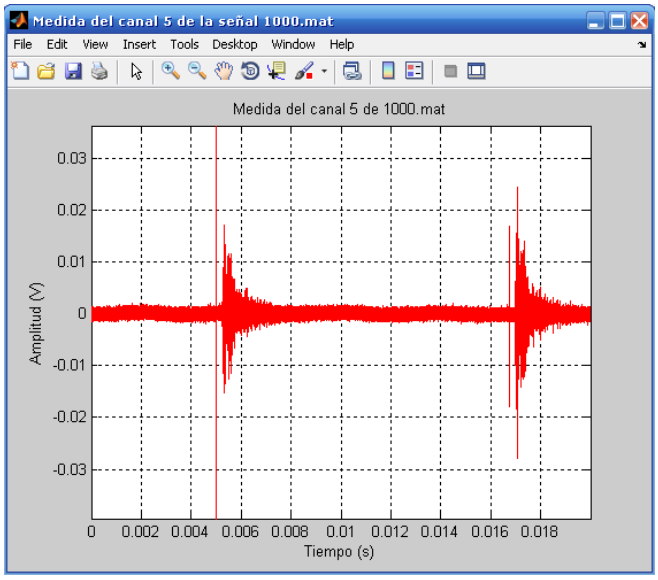


Figura 115.- Canal de la adquisición mostrado de forma individual

10.6 Motor de búsqueda

Una vez creado y seleccionado un patrón, se procede a buscar señales.

En el menú Principal (Figura 102), pulsando el botón Activación Motor de Búsqueda se abre la ventana del motor de búsqueda (Figura 116). Este será el encargado de encontrar

señales DP, si es que las hay. Es aconsejable elegir patrón adecuado para obtener los mejores resultados posibles.

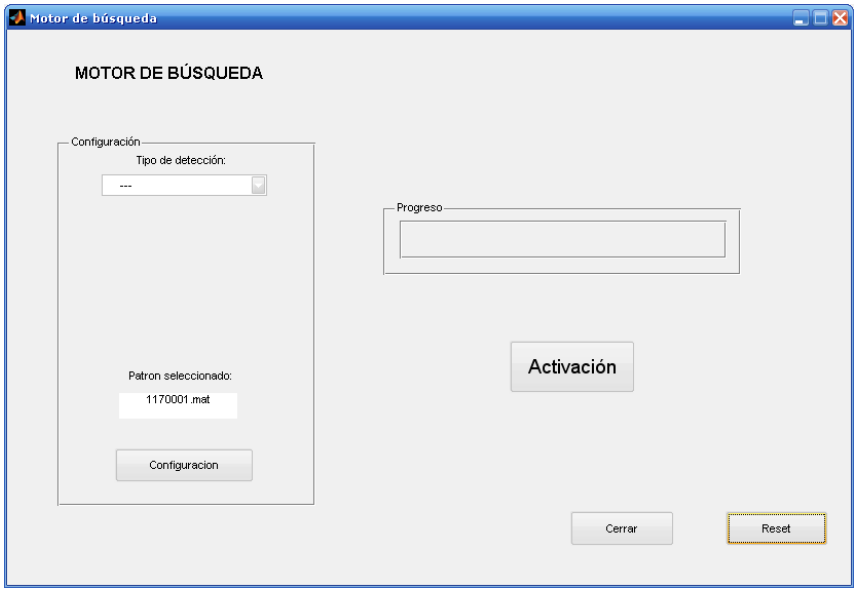


Figura 116.- Ventana del Motor de búsqueda

Se dispone de varios modos de trabajo para la búsqueda de señales DP. Estos se pueden seleccionar en el menú desplegable Tipo de selección:

- **Detección de un único canal:** Realizará la búsqueda en un rango de señales exclusivamente en un canal y empleando por tanto un único patrón. Este es el nivel más básico de búsqueda. No proporcionará información suficiente como para determinar el evento DP ya que esta búsqueda únicamente se realiza con los canales acústicos, pero si se puede tener idea del número de señales que encuentra en un canal utilizando determinado patrón.

- **Detección acústica conjunta:** Similar a la detección de un único canal, pero en este caso se analizarán todos los canales acústicos de la señal.

- **Detección con referencia temporal:** Este último modo es el que proporcionará la información de los eventos DP posibles. Analizará todos los canales acústicos del intervalo de señales. Una vez termine con las señales acústicas, intentará asociar las señales acústicas que considere válidas con la señal eléctrica y así ver cuántos eventos DP se han dado.

La ventana se irá adaptando al tipo de búsqueda que se seleccione en el menú desplegable, ocultando o mostrando los distintos campos que se deberán rellenar para introducir la información necesaria.

Seleccionado el tipo de búsqueda y rellenados los campos de intervalo de señales o canal, en el caso que sea necesario, se procede a la configuración de la búsqueda pulsando el botón Configuración.

La Configuración del Motor de Búsqueda (Figura 117) permite una configuración más profesional de la búsqueda. Se puede cambiar todas las variables que influyen en la búsqueda, como por ejemplo el nombre de la wavelet empleada en la descomposición o se puede permitir que el programa muestre ventanas que por defecto no mostrará ya que no son las que se han considerado principales.

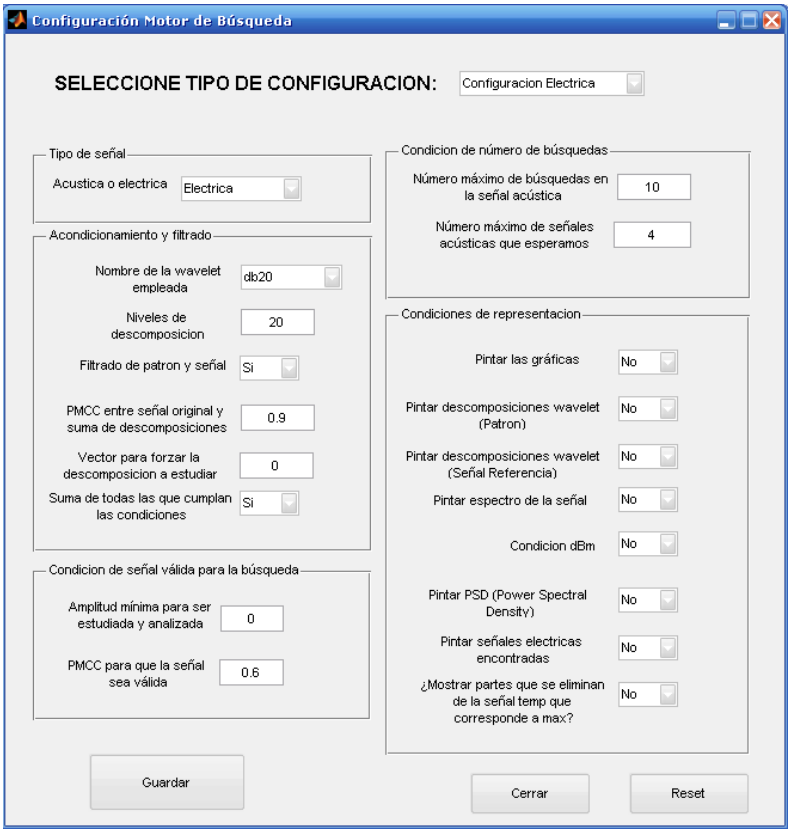


Figura 117.- Configuración para la detección

Para poder realizar cualquier búsqueda se tienen que generar dos archivos básicos de configuración. Estos son *ConfigElec.mat* y *ConfigAcus.mat*, que corresponden con la configuración eléctrica y acústica respectivamente.

Para ello se selecciona el tipo de configuración en el menú desplegable superior derecho. Se editan los campos de configuración en función de la búsqueda a realizar y se crean los archivos pulsando el botón Guardar. Si no se cambia ningún campo se guardará la configuración por defecto que se muestra. El programa informará de la creación de los archivos de configuración.

Realizada la configuración y rellenados los campos, se procederá a activar el motor de búsqueda pulsando el botón Activación (Figura 116).

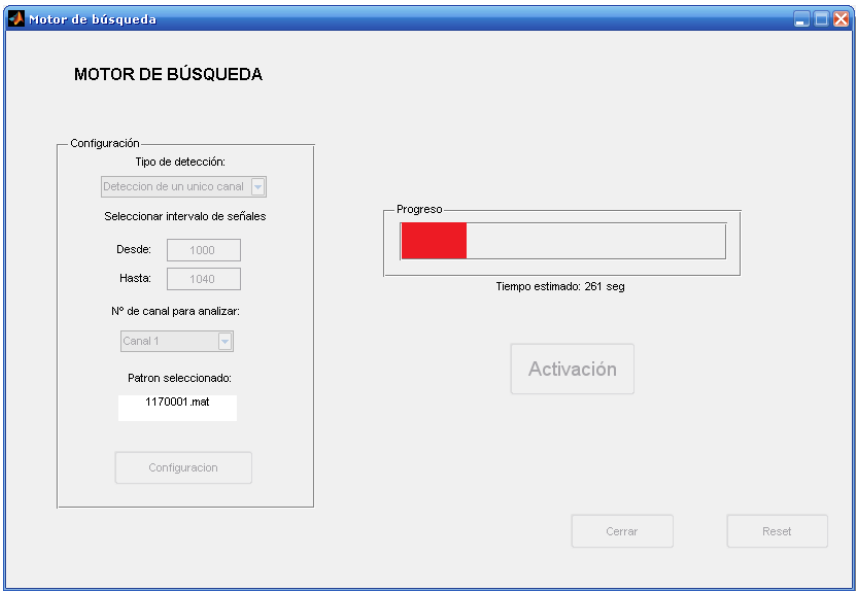


Figura 118.- Progreso de la detección en curso

La barra de progreso indica en que parte del proceso se encuentra la búsqueda, y realiza un cálculo estimado del tiempo restante (Figura 118). Hasta que no se ha terminado la búsqueda, ningún botón estará activo para evitar bloqueos involuntarios del programa al pulsar por error alguno de ellos.

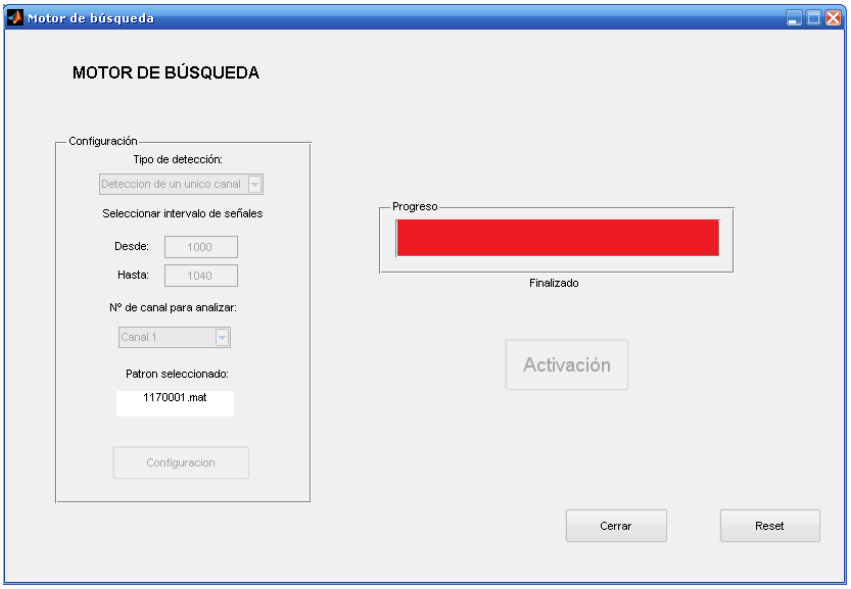


Figura 119.- Proceso de detección finalizado

Una vez termine la búsqueda, la pantalla indicará el fin del proceso (Figura 119) y se habrán creado los archivos correspondientes a cada búsqueda. Tras la búsqueda, se podrán analizar los datos obtenidos.

10.7 Visualización de eventos

Esta primera herramienta se encuentra en el menú Principal (Figura 102). Pulsando el botón Visualización Eventos se abre la ventana (Figura 120) que permite la visualización de los eventos encontrados. La ventana se configura en función de la búsqueda que se haya realizado previamente.

En la ventana se observa un menú desplegable con la etiqueta Seleccione señal, donde se cargarán los nombres de las señales que se han analizado.

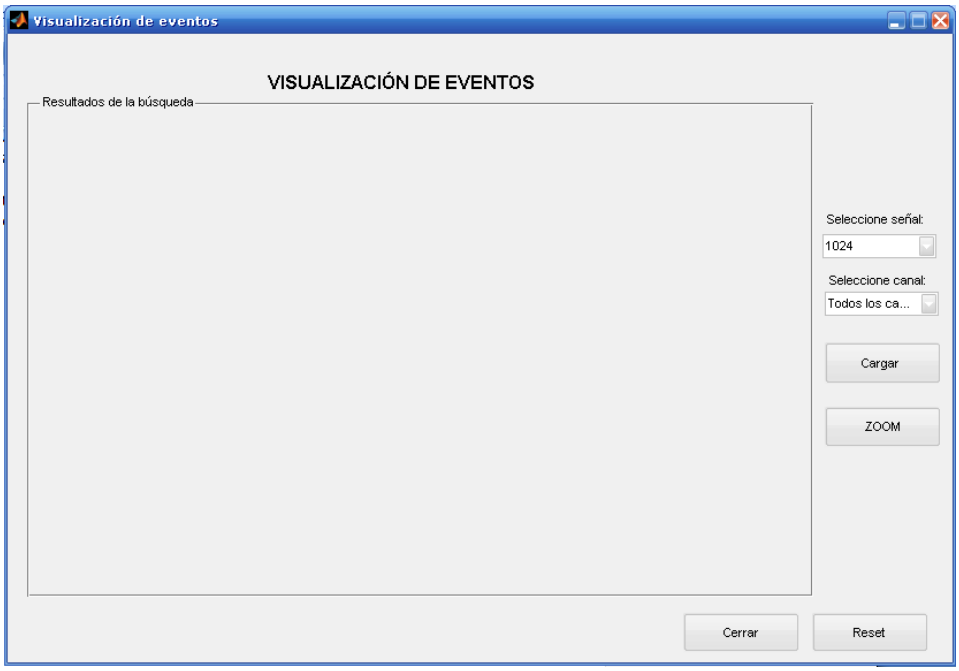


Figura 120.- Visualización de los eventos encontrados tras la detección

Al haber distintos tipos de búsquedas, se obtienen distintas ventanas y distintas gráficas. Para el caso de Detección en un único canal, se cargará en el menú desplegable los nombres de las señales, y se mostrará el canal analizado de la señal seleccionada (Figura 121). Aparece la señal original en color azul, con las señales encontradas sobrepuestas en color verde.

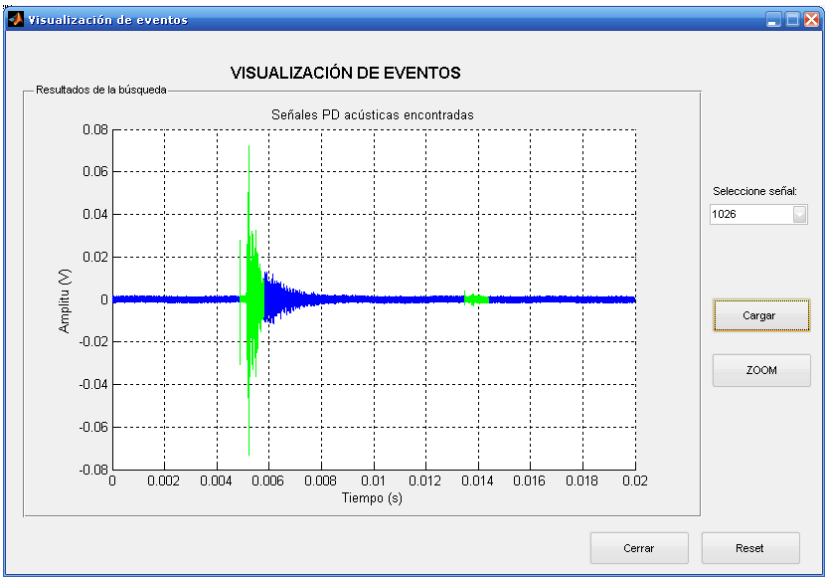


Figura 121.- Visualización de eventos para la detección en un solo canal

De la misma manera, para la detección acústica conjunta se mostrará el mismo menú desplegable para las señales y otro para la selección del canal a visualizar (Figura 122).

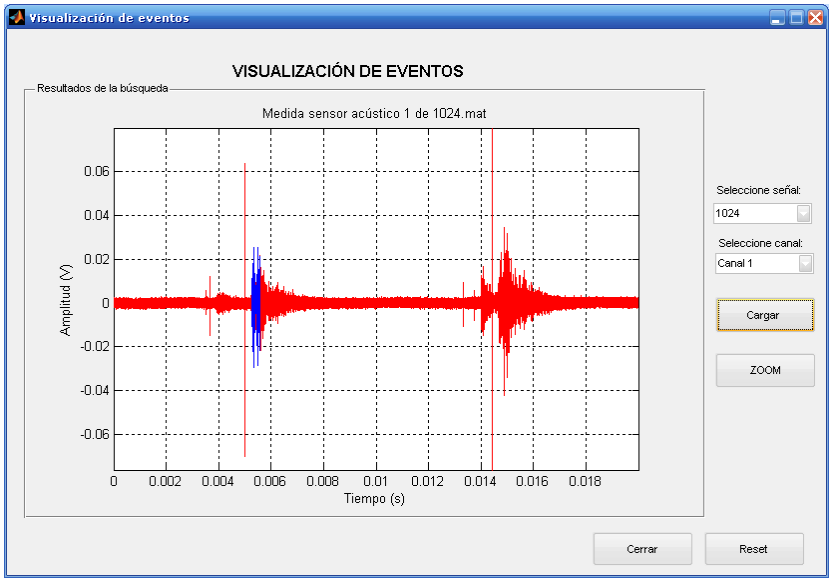


Figura 122.- Visualización de eventos para una detección global conjunta

Para el último tipo de búsqueda, la detección con referencia temporal, se mostrarán todos los canales acústicos y eléctricos de la señal seleccionada en color rojo, y pinta en azul las señales asociadas al evento DP que considera válido (Figura 123).

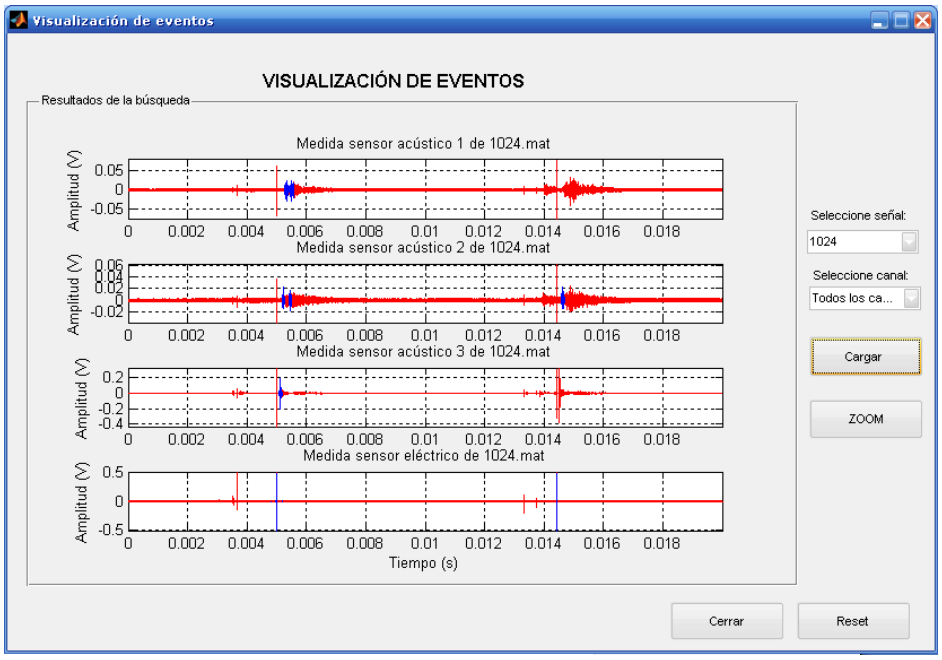


Figura 123.- Visualización de eventos para una detección con referencia temporal

La ventana dispone de una herramienta de Zoom, para aumentar o destacar las zonas de mayor interés.

Una vez realizada la selección, se ajustará la gráfica a los límites seleccionados (Figura 124). Para deshacer esta selección habrá que pulsar el botón *Reset*.

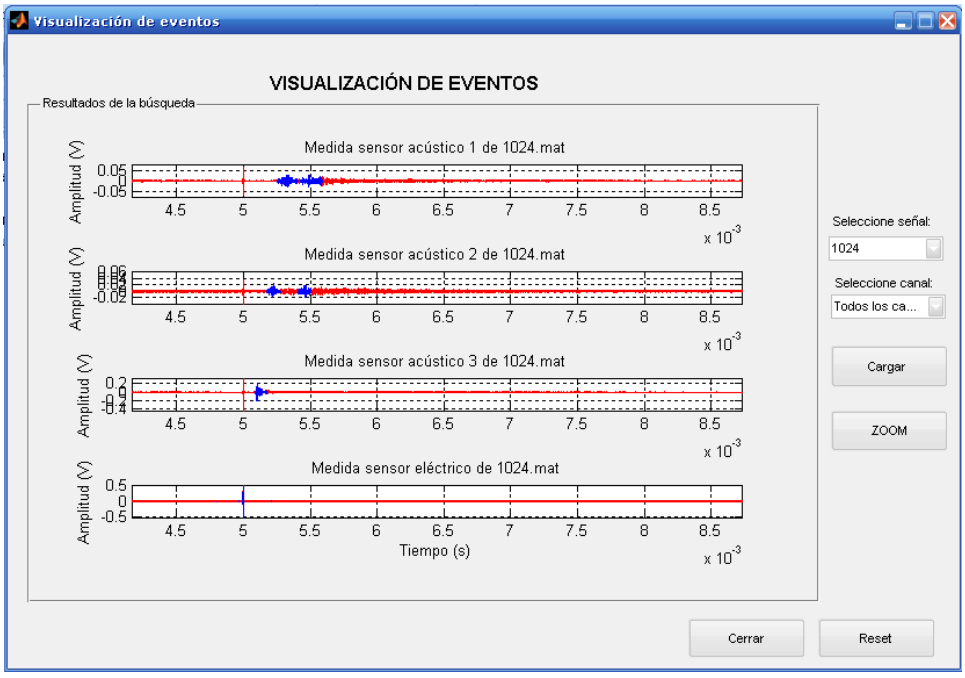
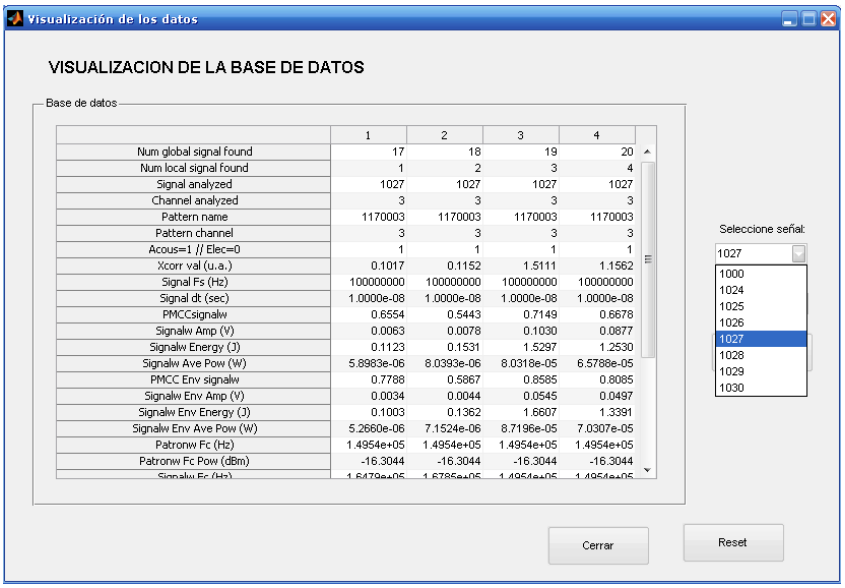


Figura 124.- Evento encontrado aumentado tras realizarle el zoom

10.8 Visualización de la base de datos

La segunda herramienta se encuentra también en el menú Principal (Figura 102). Pulsando el botón Visualización Base de Datos se abre la ventana (Figura 125) que permite la visualización de las señales encontradas. La ventana se configura en función de la búsqueda que se haya realizado previamente.

En la ventana se observa un menú desplegable con la etiqueta Seleccione señal, donde se cargarán los nombres de las señales que se han analizado y otro menú desplegable para la selección del canal (Figura 126). Tras hacer la selección de los datos, pulsar Cargar y los datos se mostrarán en la tabla.



	1	2	3	4
Num global signal found	17	18	19	20
Num local signal found	1	2	3	4
Signal analyzed	1027	1027	1027	1027
Channel analyzed	3	3	3	3
Pattern name	1170003	1170003	1170003	1170003
Pattern channel	3	3	3	3
Acous=1 // Elec=0	1	1	1	1
Xcorr val (u.a.)	0.1017	0.1152	1.5111	1.1562
Signal Fs (Hz)	100000000	100000000	100000000	100000000
Signal dt (sec)	1.0000e-08	1.0000e-08	1.0000e-08	1.0000e-08
PMCCsignalw	0.6554	0.5443	0.7149	0.6678
Signalw Amp (V)	0.0063	0.0078	0.1030	0.0877
Signalw Energy (J)	0.1123	0.1531	1.5297	1.2530
Signalw Ave Pow (W)	5.8983e-06	8.0393e-06	8.0318e-05	6.5788e-05
PMCC Env signalw	0.7788	0.5867	0.8585	0.8085
Signalw Env Amp (V)	0.0034	0.0044	0.0545	0.0497
Signalw Env Energy (J)	0.1003	0.1362	1.6607	1.3391
Signalw Env Ave Pow (W)	5.2660e-06	7.1524e-06	8.7196e-05	7.0307e-05
Patronw Fc (Hz)	1.4954e+05	1.4954e+05	1.4954e+05	1.4954e+05
Patronw Fc Pow (dBm)	-16.3044	-16.3044	-16.3044	-16.3044

Figura 125.- Selección de la adquisición de la que se desea la información de los eventos encontrados

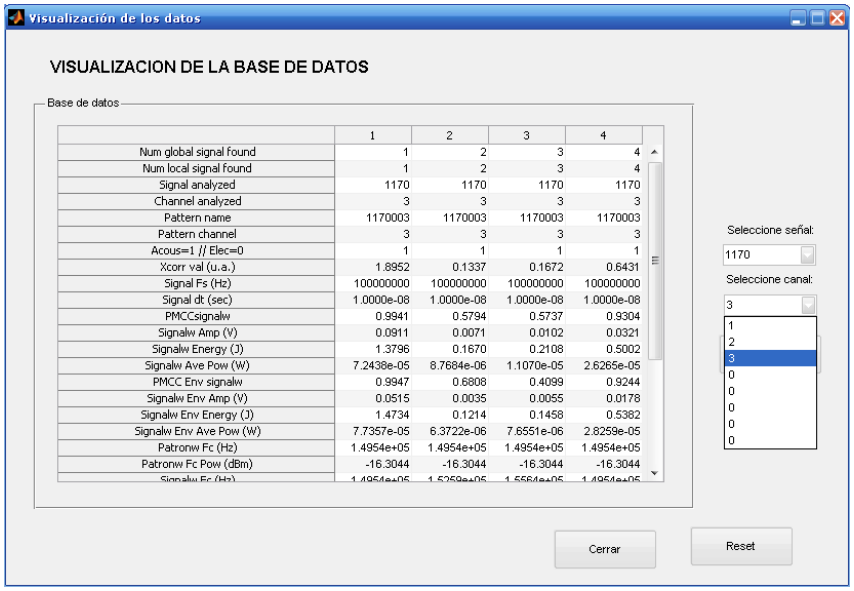


Figura 126.- Selección de canal de la adquisición de la que se desea la información de los eventos encontrados

10.9 Herramientas estadísticas

Pulsando el botón Herramientas Estadísticas en el menú Principal, se abre la ventana de Herramientas Estadísticas (Figura 127). En la ventana se cargarán los datos en función de la búsqueda previa que se haya realizado.

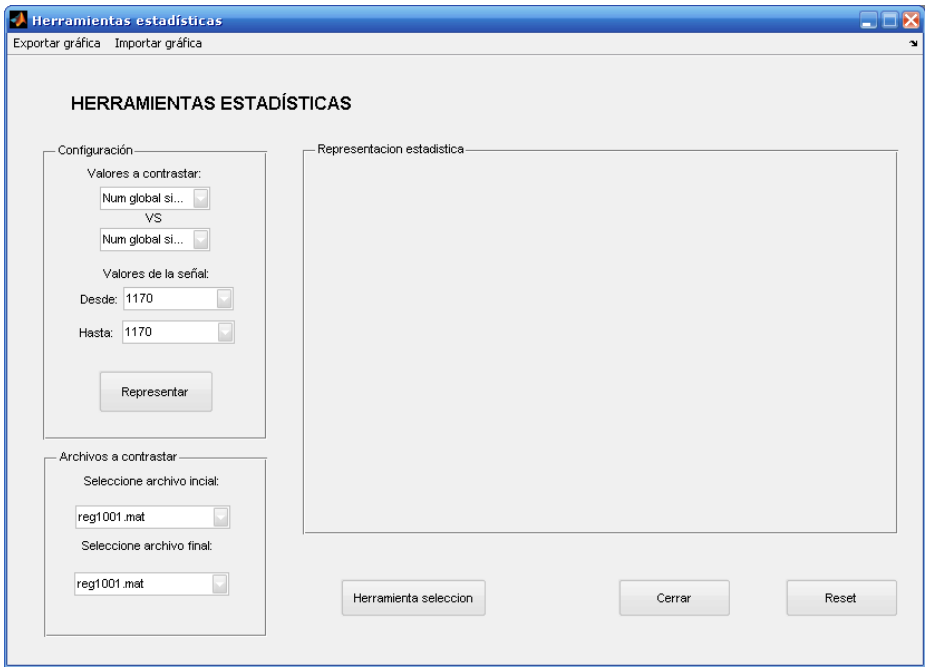


Figura 127.- Selección de datos a comparar en la herramienta estadística

En la configuración se puede elegir los valores a contrastar de los menús desplegables, así como el intervalo de señales a mostrar.

Cuando la búsqueda implica varios canales, también se podrán comparar los distintos canales, o lo que es lo mismo, los archivos generados por las búsquedas.

Una vez configurada la ventana y pulsando el botón Representar se representarán los datos seleccionados (Figura 128).

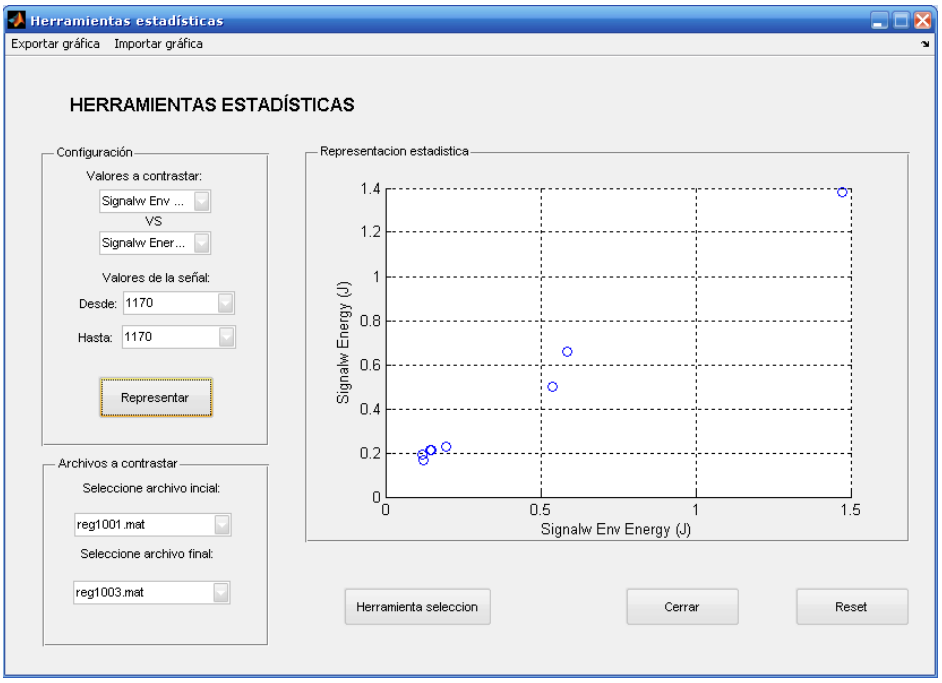


Figura 128.- Representación de datos en la herramienta estadística

Una vez pintada la gráfica, se puede activar la herramienta de selección. Para ello se debe pulsar el botón Herramienta selección. Al pulsar el botón se abre una ventana con la misma gráfica representada y se activa la herramienta de selección de puntos. Se debe seleccionar un conjunto de puntos como se muestra en la Figura 129.

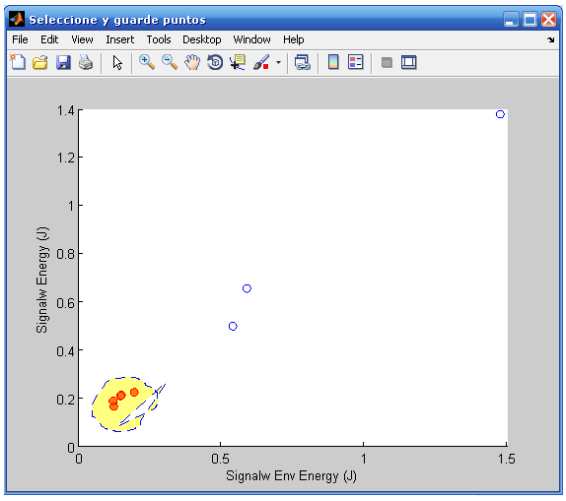


Figura 129.- Selección de puntos de interés en la herramienta estadística

Una vez realizada la selección, el programa procede a pedir el nombre del archivo que se va a generar con los puntos asociados a las descargas parciales (Figura 130).

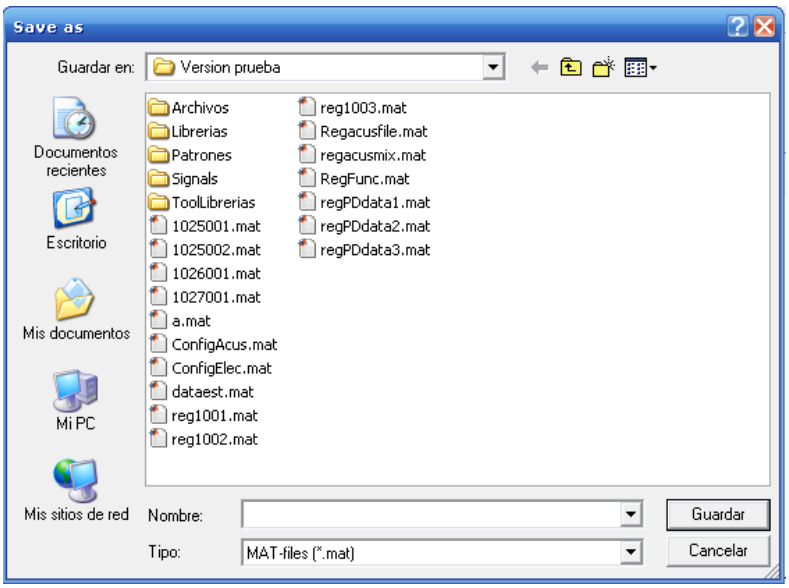


Figura 130.- Guardar selección de puntos de interés

El archivo generado contendrá la variable *selectest* (Figura 131) que a su vez contiene los siguientes datos:

selectest <9x2 cell>

	1	2
1	'Signals'	'1170'
2	'SignalPos'	[1,1]
3	'PDidx'	[2;3;4;6;7]
4	'Signalw Env...	'Signalw Ene...
5	0.1492	0.2141
6	0.1196	0.1902
7	0.1939	0.2260
8	0.1214	0.1670
9	0.1458	0.2108

Figura 131.- Variable almacenada (*selectest*) que contiene la información sobre los puntos almacenados

- ***Signals*:** Nombre de las señales que se han analizado en la búsqueda.

- ***SignalPos*:** Rango de señales entre las que se ha utilizado la herramienta de selección de datos. La información que contiene es el número de la primera y la última señal que se ha analizado de la lista de señales de la celda *Signals*.

- ***PDidx*:** Índices asociados a los eventos DP correspondientes a las señales que se han seleccionado.

- ***selectest*(4,1) y *selectest*(4,2):** Nombre del tipo de datos que se representan el los ejes x e y respectivamente. Debajo de ellos se escriben los datos numéricos de la selección, cada fila correspondiendo a uno de los *PDidx* asociado.

Existe la posibilidad de exportar o importar las gráficas en formato JPG. Para ello se seleccionará en los menús superiores de la pantalla (Figura 132) la opción de exportar o importar. Pinchando en la opción Exportar en *.jpg se exportará la gráfica actual.

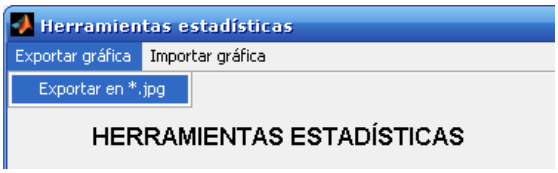


Figura 132.- Exportar gráficas en formato JPG

Una vez nombrado el archivo que se va a generar, se obtiene una imagen de la gráfica en formato JPG como se muestra en la Figura 133.

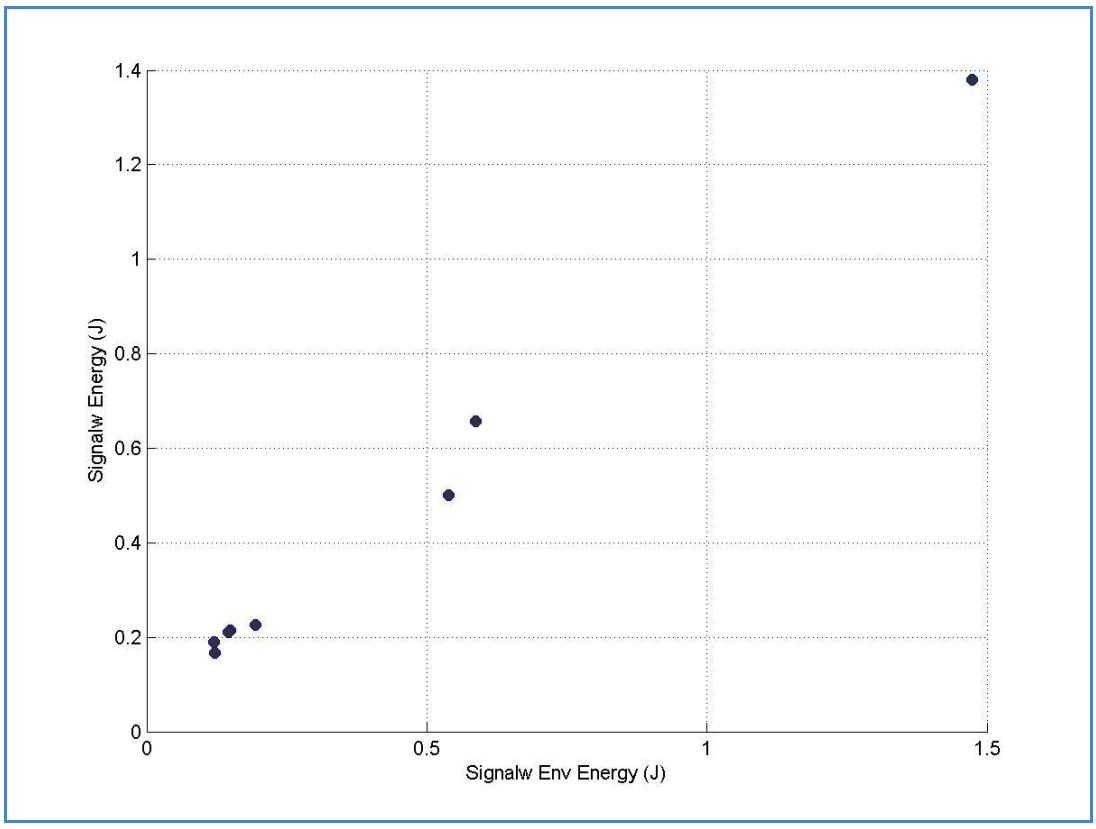


Figura 133.- Imagen de la herramienta estadística exportada en .JPG

La importación de las imágenes se realiza del mismo modo, pulsando en el menú Importar gráfica, seleccionando el archivo y aceptando.

10.10 Localización espacial

La herramienta Localización solo está disponible para búsquedas con referencia temporal. Consta del panel Tipo de localización (Figura 134) con un menú desplegable donde se selecciona entre los posibles tipos de localización: Planos, Tridimensional, Histogramas y Avanzado.

El panel de Configuración permite configurar las opciones disponibles para cada tipo de localización. Hay una serie de opciones que se podrán modificar en todos los tipos de localizaciones como son la *Vsound min*, que será la velocidad del sonido a la que se propagan las ondas acústicas o la variación de la altura del corte, así como los valores máximo y mínimo de la altura de corte de la localización. Todas las localizaciones muestran como primer valor el correspondiente a la Z min. Pulsando en la barra de *slider*, mostrará el siguiente corte teniendo en cuenta el Z paso o variación de Z. La barra de *slider* llegará al máximo y representará el valor de Z máx.

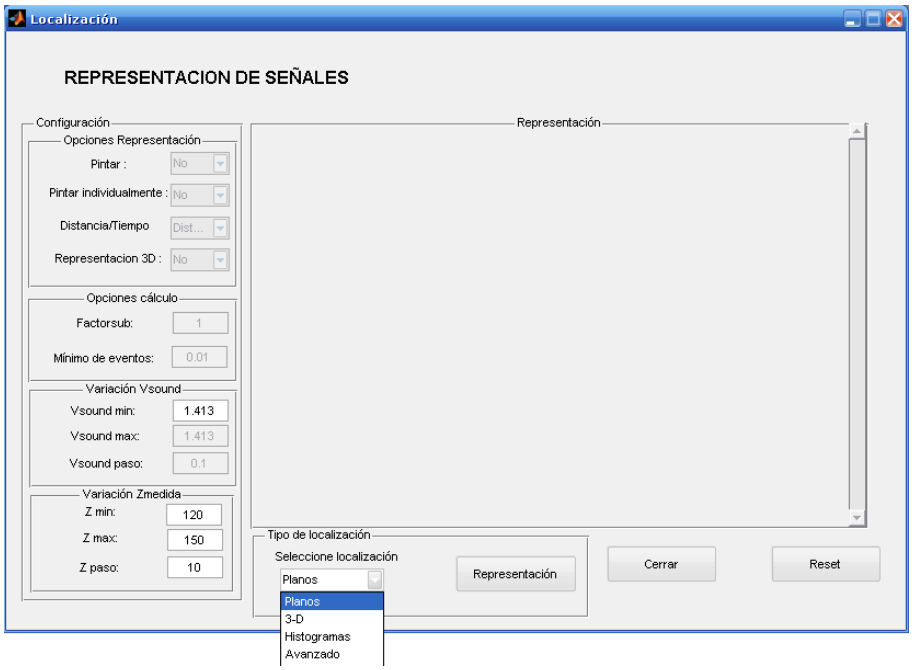


Figura 134.- Ventana de la herramienta de localización

A continuación se describen los distintos tipos de localización.

10.10.1 Localización por planos

La localización por planos muestra una sección transversal de la cuba en la que se representan las circunferencias correspondientes a la distancia a la que los sensores detectaron zonas de alta probabilidad de evento DP (Figura 135). Pulsando la barra de slider se podrán visualizar planos superiores o inferiores a Z variando la altura de la cuba (Figura 136), variando también en este caso la representación gráfica.

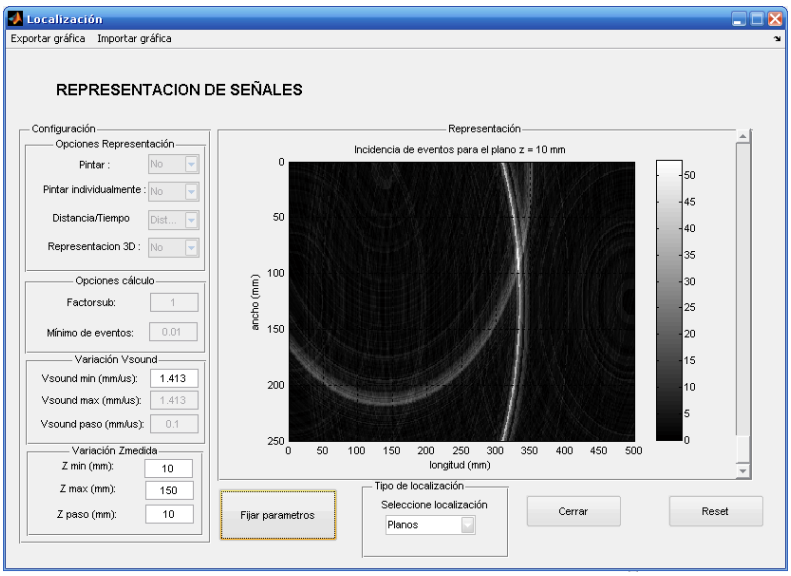


Figura 135.- Localización por planos en un punto inicial

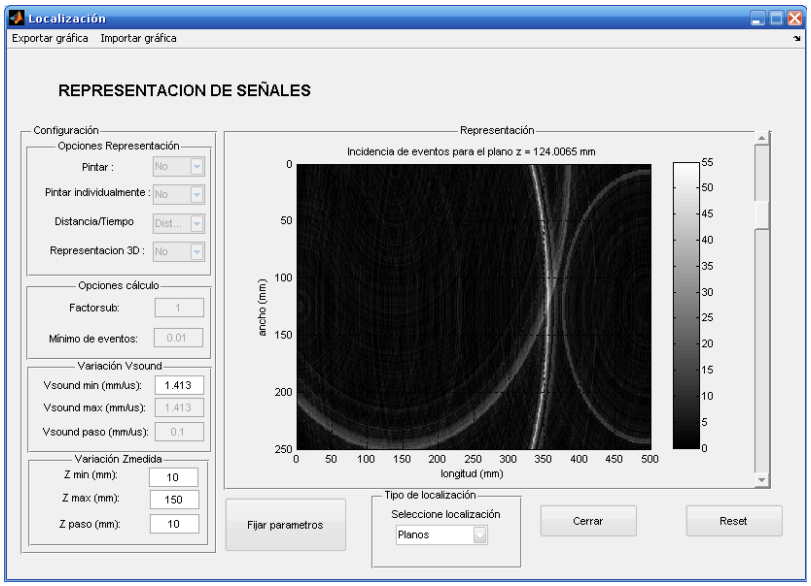


Figura 136.- Localización por planos donde se ha desplazado por la cuba mediante la posición de la barra de deslizamiento

10.10.2 Localización tridimensional

La localización tridimensional permite tener una vista 3D de la cuba (Figura 137). Este tipo de localización representa la densidad de eventos que se han dado en ciertas posiciones espaciales, siendo el punto de mayor densidad de eventos el que más probabilidad tiene de ser la fuente del evento.

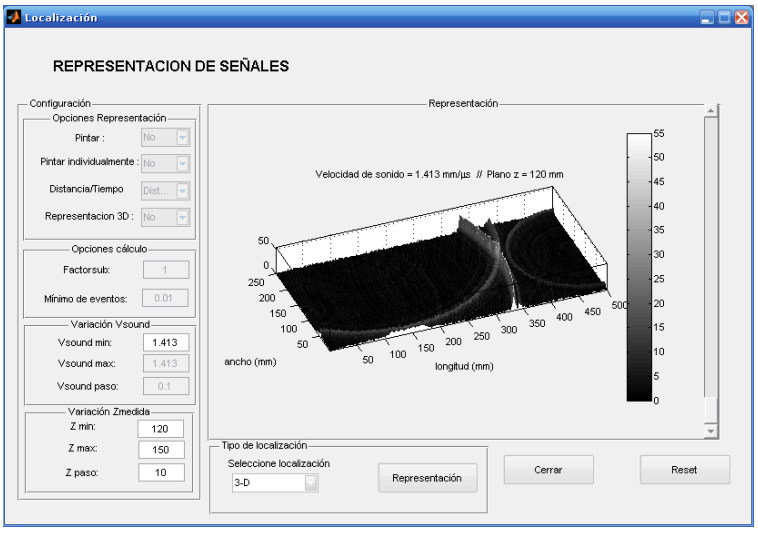


Figura 137.- Localización 3D

El esquema 3D se puede mover en el espacio pinchando sobre él (Figura 138También se desplazará en la dirección Z de la cuba con la barra *slider*. Cada movimiento de la barra desplaza Z paso unidades, hasta un valor máximo de Z máx.

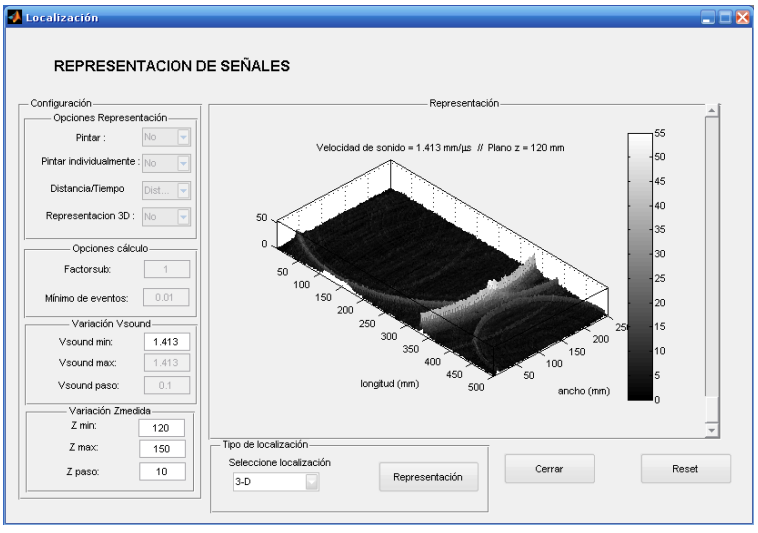


Figura 138.- Localización 3D orientado desde otra perspectiva

10.10.3 Localización por histogramas

La localización de tipo histogramas muestra los histogramas de los canales acústicos, así como una representación en el plano de la incidencia de eventos.

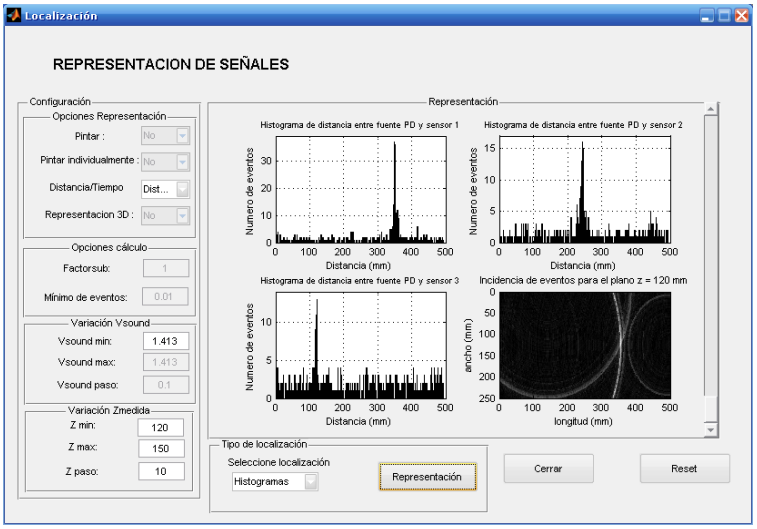


Figura 139.- Localización por histogramas basados en la distancia a la que se detectó el evento

Se pueden configurar las unidades del histograma, mostrando la distancia entre la fuente DP y los sensores (Figura 139) o el retraso entre la detección eléctrica y el sensor (Figura 140).

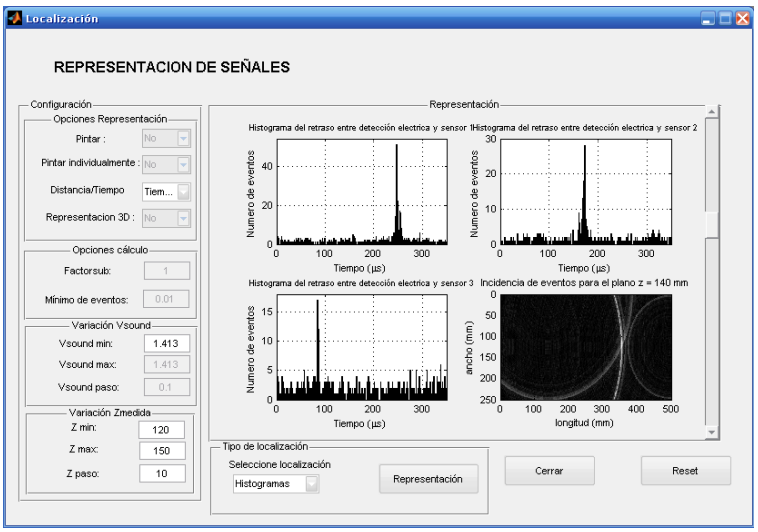


Figura 140.- Localización histogramas basados en el tiempo que tarda en llegar la señal provocada por el evento al sensor correspondiente

10.10.4 Localización avanzada

La localización avanzada permite al usuario configurar (Figura 141) todos los aspectos de la representación (respetando la funcionalidad inicial de las funciones suministradas). Al ser una localización más experta se abrirán de forma individual cada uno de los cortes en Z que se vayan a procesar. Permite también variar ciertas opciones de cálculo cuya explicación implica un conocimiento más experto de la localización.

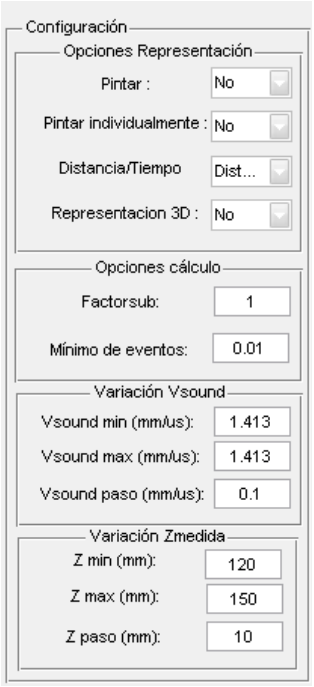


Figura 141.- Configuración de la localización avanzada

Al igual que la herramienta estadística, existe la posibilidad de exportar o importar las gráficas en formato JPG. Para ello se seleccionará la opción deseada pinchando en los menús superiores de la pantalla (Figura 142).

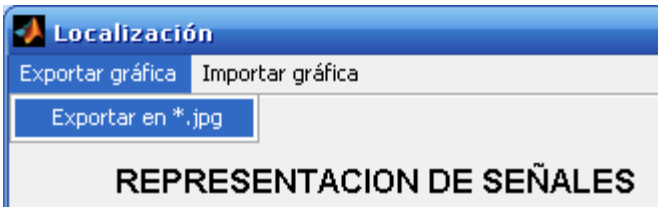


Figura 142.- Exportar gráficas en formato JPG

Una vez nombrado el archivo que se va a generar, se obtiene una imagen de la gráfica en formato JPG como se muestra en la Figura 143.

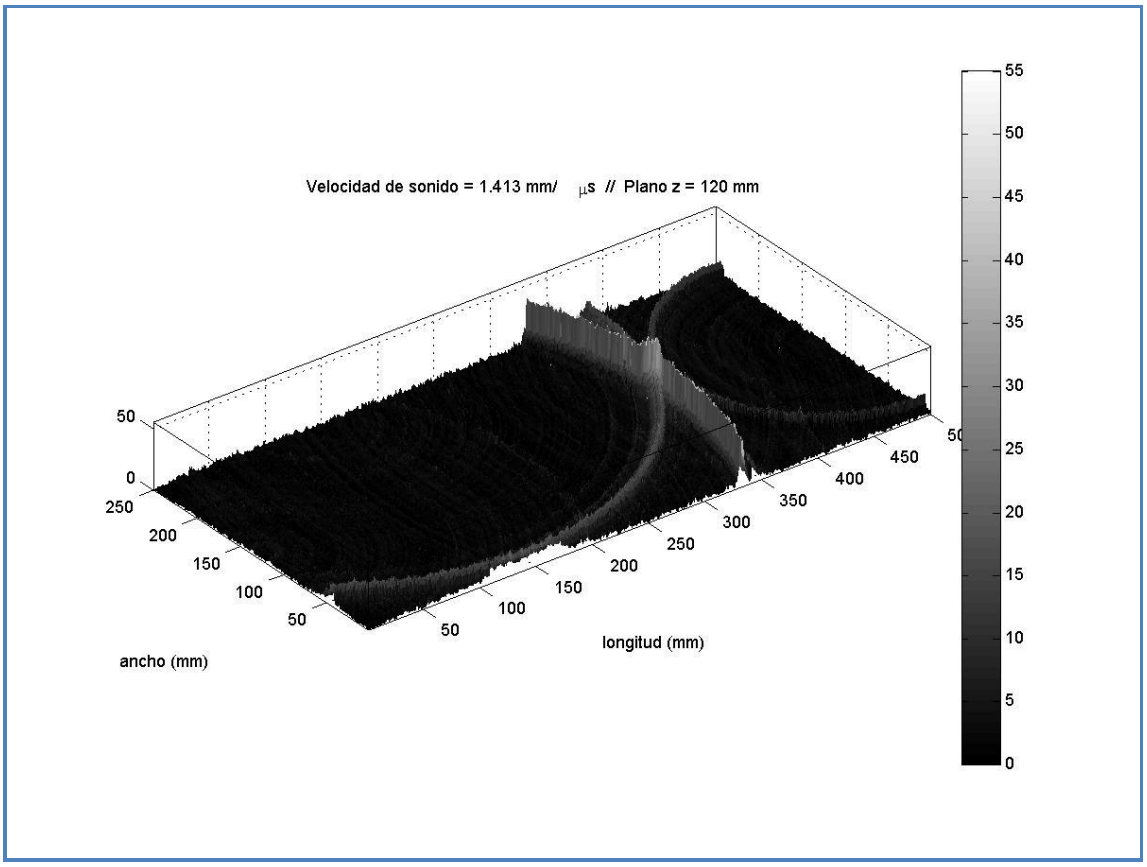


Figura 143.- Imagen de localización exportada en formato *.jpg*

.

La importación de las imágenes se realiza del mismo modo, pulsando en el menú Importar gráfica, seleccionando el archivo y aceptando.

10.11 Guardar los datos

Cuando se termine de realizar las búsquedas, y se pulse el botón cerrar, el programa preguntará si se desea guardar los datos. Pulsando Sí, el programa no borrará los archivos generados (Tabla 8) y de esa manera se podrá trabajar con ellos cuando se vuelva a encender el programa. Pulsando No, se borrarán todos los registros generados.
Durante la ejecución del programa, este guarda todos los archivos generados en el directorio raíz /PDtool. Estos archivos están resumidos en la de archivos generados (Tabla 8).

Archivo	Proceso que lo genera
Patrones (Ej: "1170003.mat")	Generado al crear un patrón.
Archivos de selección de datos.	Generados al seleccionar puntos con la herramienta de selección en la herramienta estadística.
ConfigAcus.mat y ConfigElec.mat	Generados antes de realizar una búsqueda

	con la ventana Configuración del motor de búsqueda.
Gráficas en formato JPG	Generadas al exportarlas en las ventanas de herramienta estadística y localización.
Registros de señales encontradas (Ej: "reg1001.mat")	Generados tras cualquier tipo de búsqueda.
Registros de DP (Ej : "regPDdata1.mat")	Generados tras la búsqueda con referencia temporal.
Regacusfile.mat y regacusmix.mat	Generados tras las búsqueda con referencia temporal o búsqueda global acústica.
RegFunc.mat	Archivo propio del programa.mat

Tabla 8.- Archivos generados durante la ejecución de la aplicación

Anexo 2 : Hojas de características

11.1 Características del equipo LG E500-J.AP51B

Figura 144.- Característica del equipo empleado

11.2 Requisitos mínimos para la instalación de MATLAB R2009a

Windows

Operating Systems	Processors	Disk Space	RAM
32-bit MathWorks Products			
Windows XP Service Pack 2 or 3 Windows Server 2003 Service Pack 2 or R2 Windows Vista Service Pack 1 or 2 Windows Server 2008 Windows 7	Intel Pentium 4 and above Intel Celeron* Intel Xeon Intel Core Intel Atom** AMD Athlon 64* AMD Opteron AMD Sempron*	680 MB** (MATLAB only)	512 MB (At least 1024 MB recommended)
64-bit MathWorks Products			
Windows XP x64 Service Pack 2 Windows Server 2003 x64 Service Pack 2 or R2 Windows Vista Service Pack 1 or 2 Windows Server 2008 Windows 7	Intel Pentium 4 and above Intel Celeron* Intel Xeon Intel Core AMD64	680 MB** (MATLAB only)	1024 MB (At least 2048 MB recommended)

* Processor must support SSE2 instruction set.

** Disk space requirement varies depending on size of partition. The MATLAB installer will inform you of the hard disk space requirement for your particular partition. Installation size will be determined by the installer and can vary for NTFS and FAT formats.

Figura 145.- Requisitos de instalación de MATLAB R2009a

Anexo 3 :Prototipos de bajo nivel y resumen entrevistas

12.1 Resumen de entrevistas

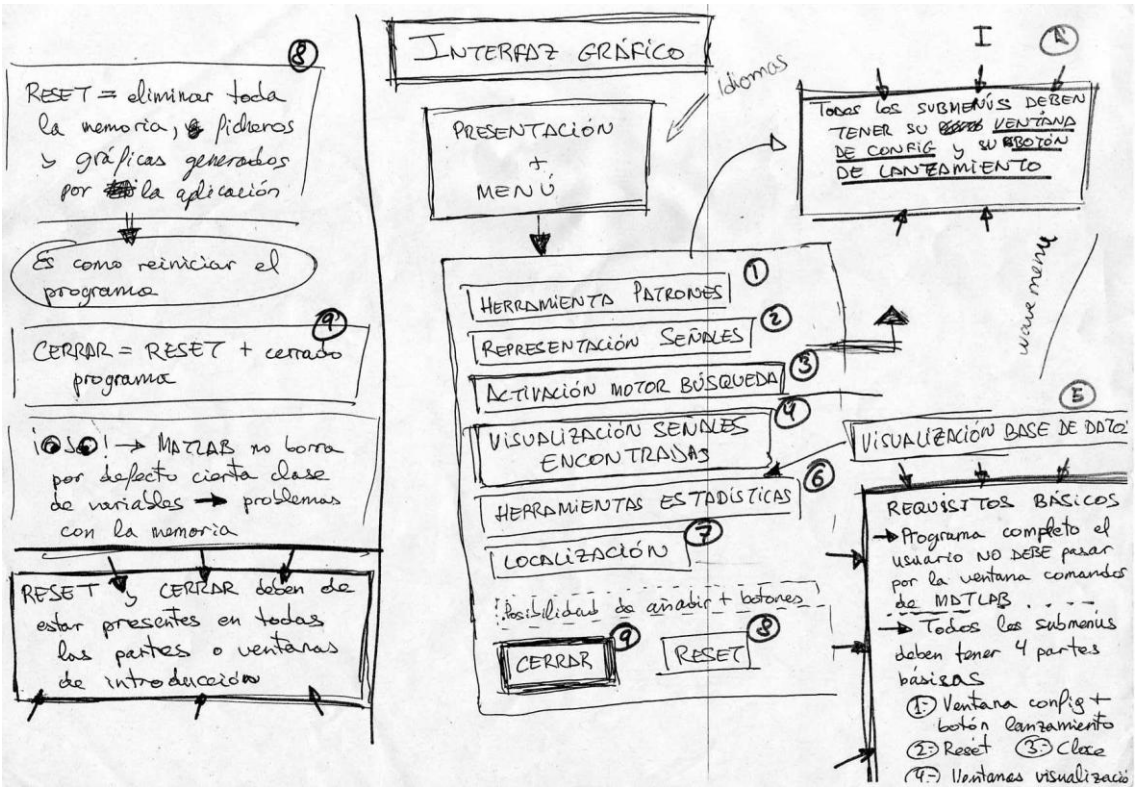


Figura 146.- Resumen de entrevistas, parte 1

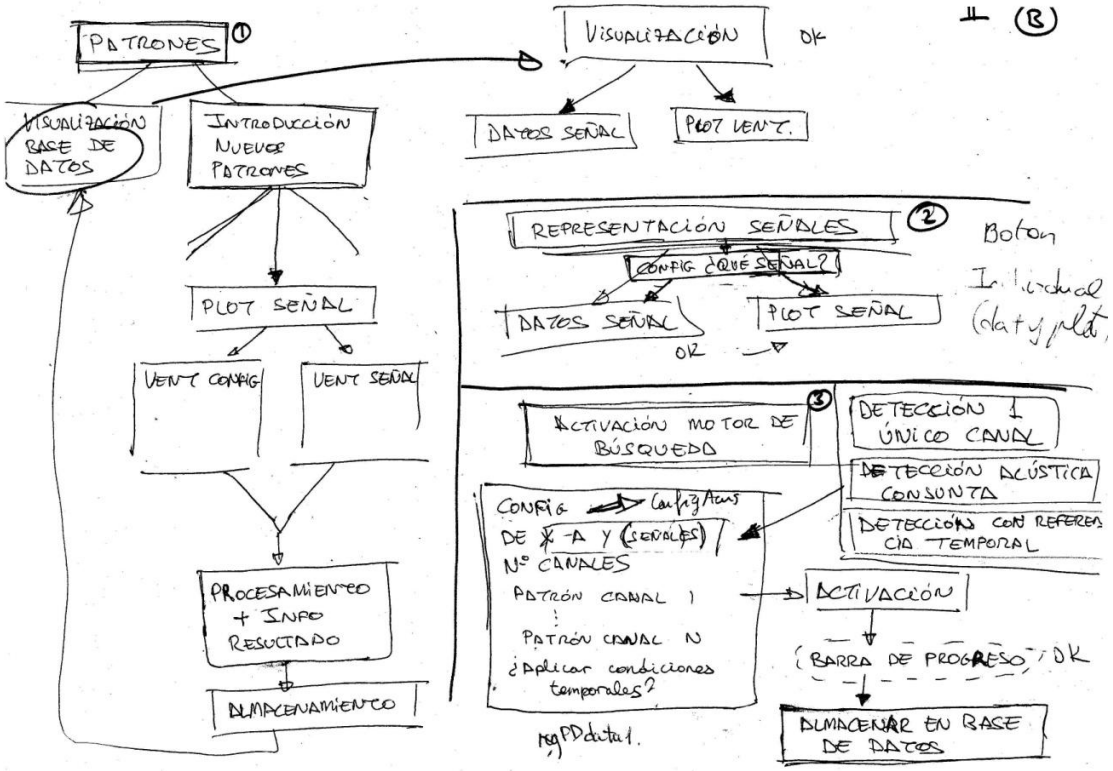


Figura 147.- Resumen de entrevistas, parte 2

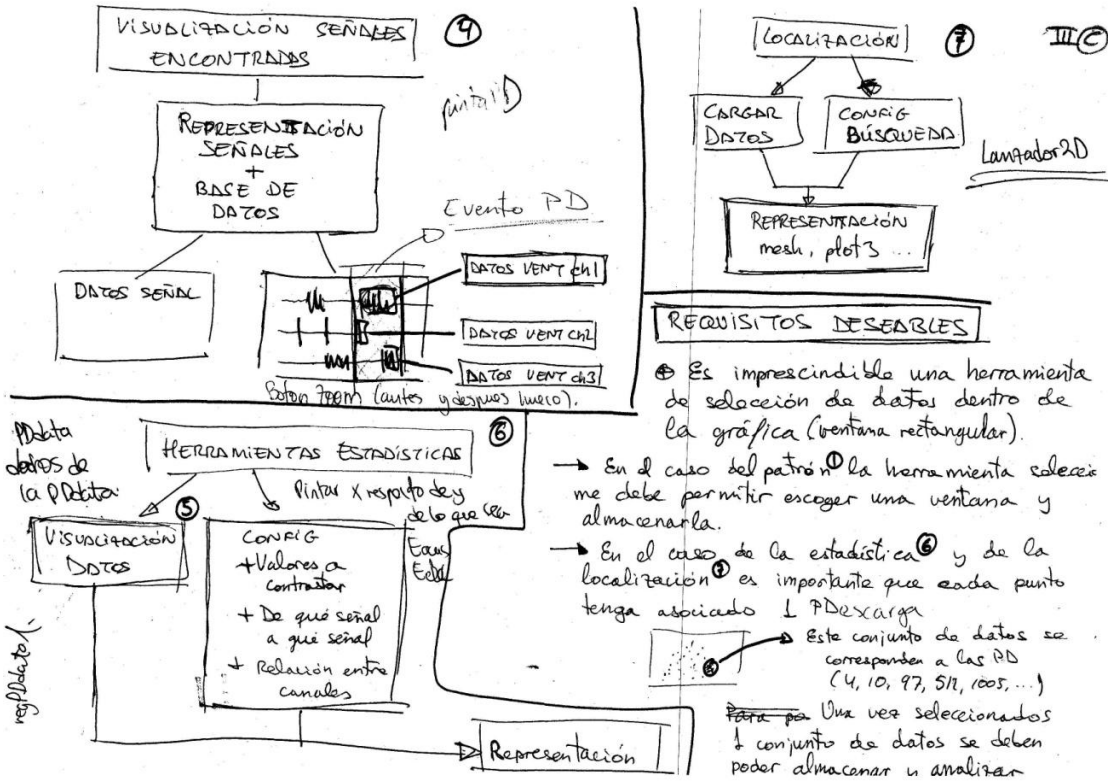


Figura 148.- Resumen de entrevistas, parte 3

12.2 Prototipos de bajo nivel

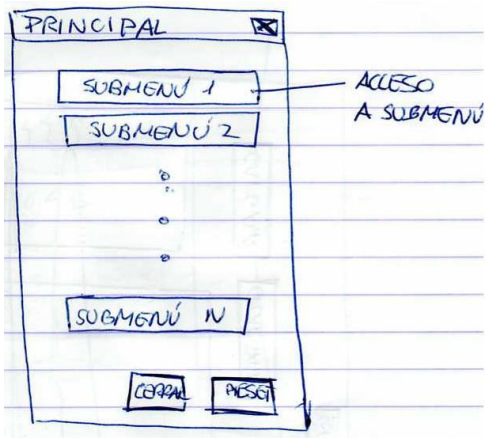


Figura 149.- Prototipo de bajo nivel del menú Principal

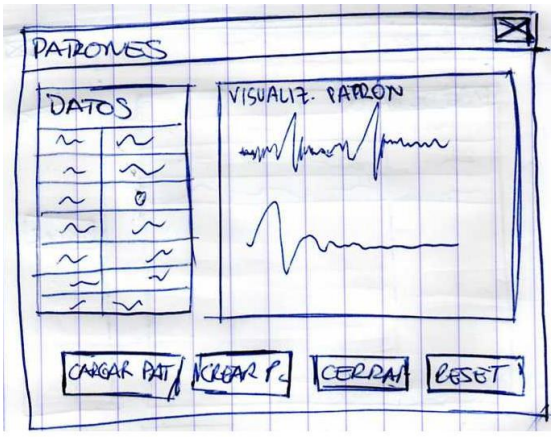


Figura 150.- Prototipo de bajo nivel de la herramienta Patrones

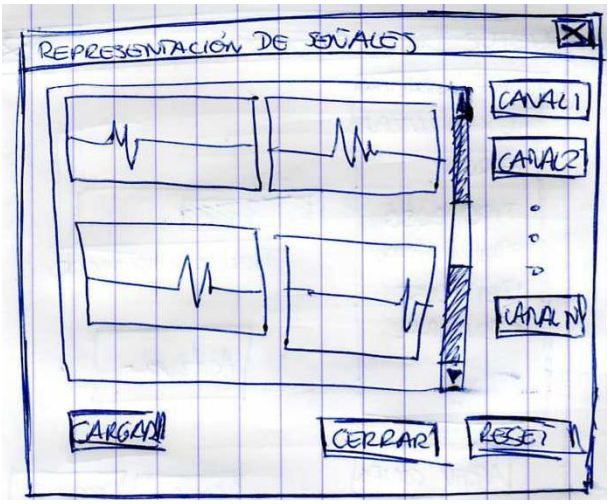


Figura 151.- Prototipo de bajo nivel de la herramienta Representación de señales

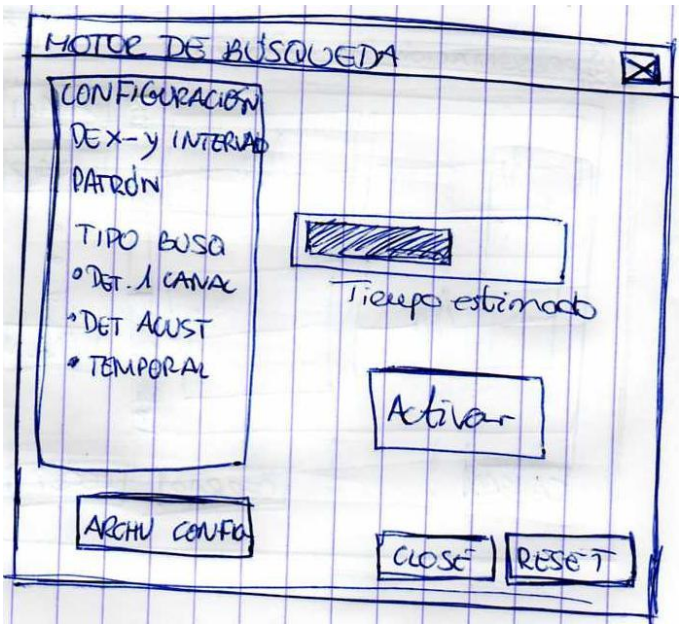


Figura 152.- Prototipo de bajo nivel de la herramienta motor

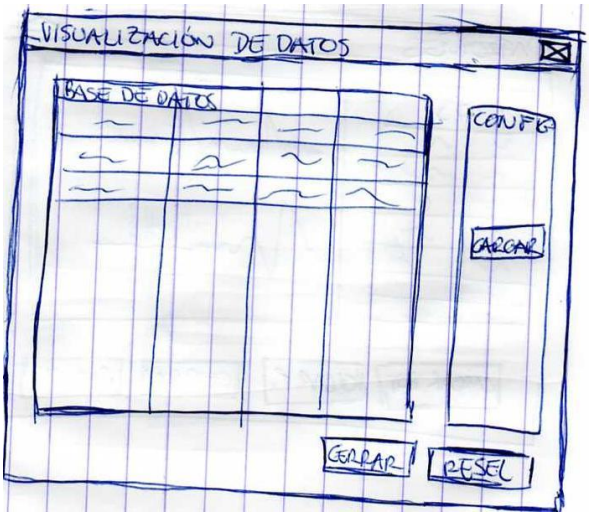


Figura 153.- Prototipo de bajo nivel de la herramienta Visualización de datos

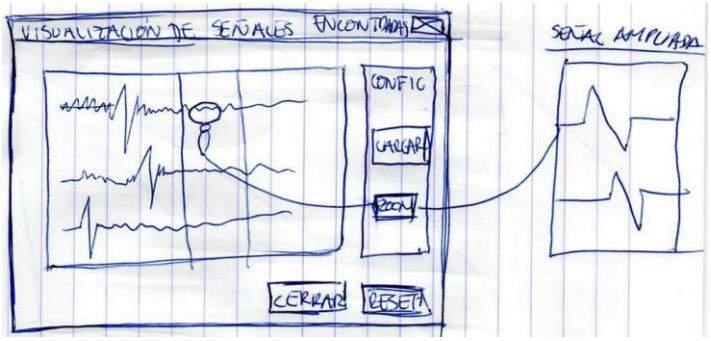


Figura 154.- Prototipo de bajo nivel de la herramienta Visualización de señales

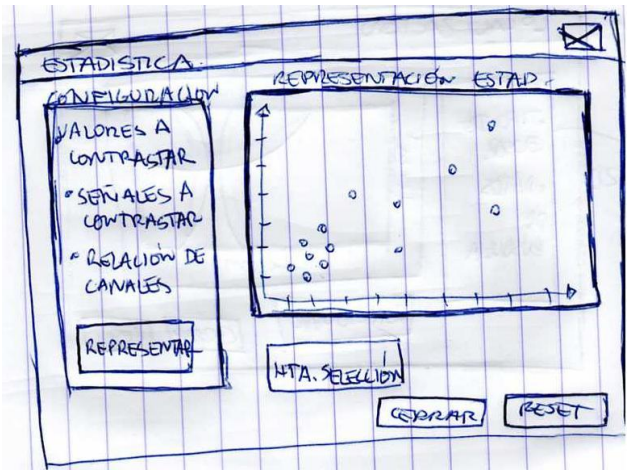


Figura 155.- Prototipo de bajo nivel de la herramienta Estadística

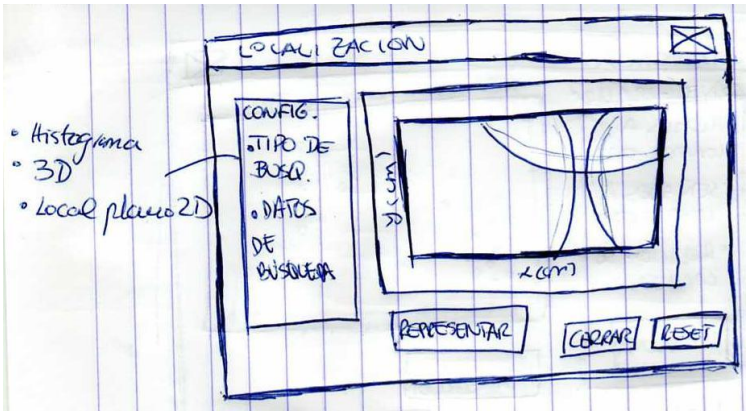


Figura 156.- Prototipo de bajo nivel de la herramienta Localización

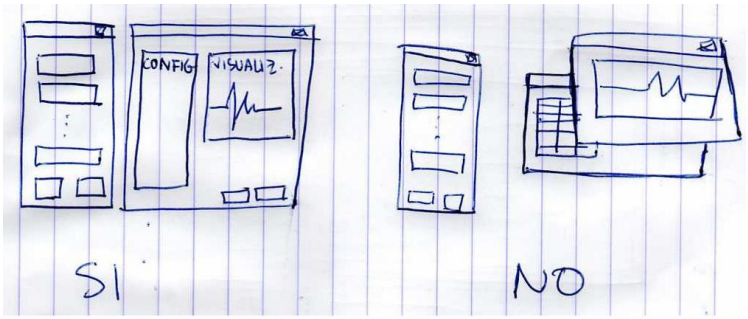


Figura 157.- Condición de diseño para evitar excesivas ventanas